# Surrogate upper bound sets
for
bi-objective bi-dimensional binary knapsack problems

Audrey Cerqueus[†]*, Anthony Przybylski, Xavier Gandibleux

Université de Nantes
LINA UMR CNRS 6241[†]
UFR sciences – 2 Rue de la Houssinière BP 92208
F-44322 Nantes Cedex 03 – France
+33 (0)2 51 12 58 48

Abstract

The paper deals with the definition and the computation of surrogate upper bound sets for the bi-objective bi-dimensional binary knapsack problem. It introduces the Optimal Convex Surrogate Upper Bound set, which is the tightest possible definition based on the convex relaxation of the surrogate relaxation. Two exact algorithms are proposed: an enumerative algorithm and its improved version. This second algorithm results from an accurate analysis of the surrogate multipliers and the dominance relations between bound sets. Based on the improved exact algorithm, an approximated version is derived. The proposed algorithms are benchmarked using a dataset composed of three groups of numerical instances. The performances are assessed thanks to a comparative analysis where exact algorithms are compared between them, the approximated algorithm is confronted to an algorithm introduced in a recent research work.

Keywords: combinatorial optimization, multiple objective programming, bi-dimensional binary knapsack problem, surrogate relaxation, bound sets.

1. Introduction

1.1. Problem formulation and characteristics

Given a set of items $J = \{1, \dots, n\}$ and a set of dimensions $I = \{1, \dots, m\}$, we associate to each dimension $i \in I$ a capacity $\omega_i \in \mathbb{N}_1$ and a set of weights $w_{ij} \in \mathbb{N}_1$ for each item $j \in J$. Moreover, each item is available in a single copy. The *multi-objective multi-dimensional binary knapsack problem* consists in packing a subset of J into a container with limited capacity over the dimensions I. This must be done while maximizing a profit according to a set of objectives $K = \{1, \dots, p\}$. For this, a profit $c_j^k \in \mathbb{N}_1$ is associated to each item $j \in J$ according to each

*Corresponding author
Email addresses: Audrey.Cerqueus@univ-nantes.fr (Audrey Cerqueus[†]), Anthony.Przybylski@univ-nantes.fr (Anthony Przybylski), Xavier.Gandibleux@univ-nantes.fr (Xavier Gandibleux)

objective $k \in K$. The general formulation of this problem is the following.

$$
\begin{aligned}
&\max \sum_{j=1}^{n} c_j^k \, x_j && k = 1, \ldots, p \\
&s.t. \sum_{j=1}^{n} w_{ij} \, x_j \leq \omega_i && i = 1, \ldots, m \\
&x_j \in \{0, 1\} && j = 1, \ldots, n \qquad (pOmDKP)
\end{aligned}
$$

This problem has received a lot of attention, see *e.g.* Fréville (2004); Varnamkhasti (2012); Lust and Teghem (2012) for recent surveys. It is used for representing various practical problems as capital budgeting, allocating processors (Martello and Toth, 1990; Kellerer et al., 2004; da Silva et al., 2004). For example, Clausen et al. (2010) consider a special case of two-dimensional knapsack for the problem of assigning seats in a train for a group of people traveling together.

Many particular cases of this problem have been studied, and all them are $\mathcal{NP}$-hard. We will denote this problem by $(pOmDKP)$, according to the number of objectives and dimensions. We will omit the number of dimensions and/or the number of objectives in this notation whenever $p = 1$ and/or $m = 1$. In the following, we will make a particular distinction between the single-dimensional ($m = 1$) and the multi-dimensional ($m > 1$) case. Indeed, (KP) problem can be considered as an "easy" $\mathcal{NP}$-hard problem as several practically efficient methods have been proposed for its exact solution (Pisinger, 1994; Martello et al., 1999; Kellerer et al., 2004). Their theoretical time-complexity is pseudo-polynomial. $(2OKP)$ problem is also one of the most studied multi-objective combinatorial optimization problems. A number of methods have been proposed for its exact solution (Ulungu and Teghem, 1997; Visée et al., 1998; Bazgan et al., 2009; Jorge, 2010; Delort and Spanjaard, 2010). Bazgan et al. (2009); Jorge (2010) have also provided exact solution methods for $(3OKP)$ problem.

$(mDKP)$ problem is practically far more difficult than (KP) problem as pointed out by Fréville (2004). In the multi-objective case, instances of this problem are often used as a benchmark to compare metaheuristics (Zitzler and Thiele, 1999; Jaszkiewicz, 2004; Ishibuchi et al., 2009; Tricoire, 2012). However, this problem has received less attention in an exact context (Florios et al., 2010; Mavrotas et al., 2011; Lust and Teghem, 2012). To our knowledge, only one exact method has been proposed for the specific $(2O2DKP)$ case (Perederieiva, 2011; Gandibleux and Perederieiva, 2011).

In this paper, we also consider the $(2O2DKP)$ problem. The main purpose is to design and compute a tight upper bound set for this problem. Such a bound set is a crucial component when it is embedded within an algorithm aiming to enumerate a complete set of nondominated points of the $(2O2DKP)$.

1.2. Efficiency, nondominance and bound sets

The main definitions, properties, and notations of multi-objective combinatorial optimization are given now. A more complete introduction can be found in (Ehrgott, 2005).

A multi-objective combinatorial optimization problem can be formulated as

$$\max\{(z_1(x), \ldots, z_p(x)) = Cx : x \in X\}, \qquad (MOCO)$$

with a linear objective matrix $C \in \mathbb{R}^{p \times n}$, variables $x \in \mathbb{R}^n$, and the feasible set (in decision space $\mathbb{R}^n$)

$$X := \{x \in \{0, 1\}^n : Ax \leqq b, \; x \geqq 0\}.$$

Matrix A is an $m \times n$ matrix of constraints and $b \in \mathbb{R}^m$ the right hand side vector. The image of X under C, *i.e.*,

$$Y := CX := \{y = Cx \in \mathbb{R}^p : x \in X\}$$

is called the outcome set in objective space $\mathbb{R}^p$.

We assume that no feasible solution optimizes all objectives simultaneously and use the following notations for componentwise orders in $\mathbb{R}^p$. Let $y^1, y^2 \in \mathbb{R}^p$. We write $y^1 \geqq y^2$ (y^1 *weakly dominates* y^2) if $y^1_k \geq y^2_k$ for $k = 1, \ldots, p$; $y^1 \geq y^2$ (y^1 *dominates* y^2) if $y^1 \geqq y^2$ and $y^1 \neq y^2$; and $y^1 > y^2$ (y^1 *strictly dominates* y^2) if $y^1_k > y^2_k, k = 1, \ldots, p$. We define $\mathbb{R}^p_{\geqq} := \{x \in \mathbb{R}^p : x \geqq 0\}$ and analogously $\mathbb{R}^p_{\geq}$ and $\mathbb{R}^p_{>}$.

A feasible solution $\hat{x} \in X$ is called *efficient (weakly efficient)* if there does not exist $x \in X$ such that $z(x) \leq z(\hat{x})$ ($z(x) < z(\hat{x})$). If $\hat{x}$ is (weakly) efficient, then $z(\hat{x})$ is called *(weakly) nondominated.* The *efficient set* $X_E \subseteq X$ is defined as

$$X_E := \{x \in X : \nexists\ \bar{x} \in X : z(\bar{x}) \leq z(x)\},$$

and its image in objective space is referred to as the *nondominated set* $Y_N := z(X_E)$. Equivalently, Y_N can be defined by $Y_N := \{y \in Y : (y + \mathbb{R}^p_{\geqq}) \cap Y = \{y\}\}$. This concept is extended by defining $S_N := \{s \in S : (s + \mathbb{R}^p_{\geqq}) \cap S = \{s\}\}$ for an arbitrary set $S \in \mathbb{R}^p$. The exact solution of a multi-objective combinatorial optimization problem consists in determining a *complete set* of efficient solutions, *i.e.* to determine at least one efficient solution for each nondominated point.

As we consider here multi-objective combinatorial optimization problems, several classes of efficient solutions need to be distinguished. *Supported* efficient solutions are optimal solutions of a weighted sum single objective problem (Geoffrion, 1968)

$$\max\{\lambda_1 z_1(x) + \ldots + \lambda_p z_p(x) : x \in X\} \qquad (MOCO_\lambda)$$

for some $\lambda \in \mathbb{R}^p_{>}$. Their images in objective space are supported nondominated points. We use the notations X_{SE} and Y_{SN}, respectively. In order to avoid a confusion with the weights of the items of ($pOmDKP$), the weight vector $\lambda \in \mathbb{R}^p_{>}$ will be called *direction* in the following. All supported nondominated points are located on the boundary of the convex hull of Y (conv Y), *i.e.*, they are nondominated points of $(\text{conv}\, Y) - \mathbb{R}^p_{\geqq}$.

Nonsupported efficient solutions are efficient solutions that are not optimal solutions of ($MOCO_\lambda$) for any direction $\lambda \in \mathbb{R}^p_{>}$. Nonsupported nondominated points are located in the interior of the convex hull of Y. In the particular bi-objective case, nonsupported nondominated points are located in the interior of triangles the hypotenuse of which is defined by consecutive supported nondominated points with respect to z_1. The set of nonsupported efficient solutions and nondominated points are denoted respectively by X_{NE} and Y_{NN}.

Finally, we can distinguish two classes of supported efficient solutions. The set of *extremal* supported efficient solutions X_{SE1} is a subset of X_{SE} the corresponding point of which is an extreme point of conv Y. $Y_{SN1} := z(X_{SE1})$ is the set of *nondominated extreme points.* $X_{SE2} := X_{SE} \setminus X_{SE1}$ and $Y_{SN2} := Y_{SN} \setminus Y_{SN1}$ are respectively the sets of *non-extremal* supported efficient solutions and nondominated points. The computation of the set Y_{SN1} can be easily done in the bi-objective context using the algorithm by Aneja and Nair (1979), under the assumption that the corresponding single-objective problem can be solved efficiently in practice.

Bounds on the optimal value of a single-objective problem are crucial to design efficient solution methods in the single-objective case. Their generalization to the multi-objective case called *bound* *sets* (Ehrgott and Gandibleux, 2001) are used to bound Y_N. Several definitions of bound sets have been proposed, we use the definition proposed by Ehrgott and Gandibleux (2007). Contrary to the single-objective case, the definitions of upper and lower bound sets are not symmetric. A lower bound set for Y_N is generally given by a set of known feasible points filtered by dominance. Upper bound sets for Y_N (or for any subset of points in $\mathbb{N}^p$) can be obtained by far more various ways.

We introduce some additional terminology before the next definition. S is $\mathbb{R}^p_{\geqq}$-*closed* if the set $S - \mathbb{R}^p_{\geqq}$ is closed and $\mathbb{R}^p_{\geqq}$-*bounded* if there exists $s^0 \in \mathbb{R}^p$ such that $S \subset s^0 - \mathbb{R}^p_{\geqq}$.

Definition 1. *(Ehrgott and Gandibleux, 2007) Let $\bar{Y} \subset \mathbb{R}^p$. An upper bound set U for $\bar{Y}$ is an $\mathbb{R}^p_{\geqq}$-closed and $\mathbb{R}^p_{\geqq}$-bounded set $U \subset \mathbb{R}^p$ such that $\bar{Y} \subset U - \mathbb{R}^p_{\geqq}$ and $U \subset (U - \mathbb{R}^p_{\geqq})_N$.*

In solution methods, $\bar{Y}$ denote generally the set of nondominated points of a subproblem of the considered problem (*e.g.* a relaxation). The same way as the single-objective context, bound sets for $\bar{Y}$ must be as tight as possible for a reasonable computational cost. Ehrgott and Gandibleux (2007) have proposed the notion of dominance between bound sets.

Definition 2. *(Ehrgott and Gandibleux, 2007) Given two upper bound sets U_1 and U_2 for a same set $\bar{Y}$, U_1 dominates U_2 if $U_1 \subset U_2 - \mathbb{R}^p_{\geqq}$ and $U_1 - \mathbb{R}^p_{\geqq} \neq U_2 - \mathbb{R}^p_{\geqq}$.*

The dominance relation between bound sets is transitive. Nevertheless, if we can always compare bound values in the single-objective case, dominance between bound sets does not necessarily occur (Figure 1). Computing several upper bound values in the single-objective context, the smallest defines naturally the tightest bound. Proposition 1 provides a way to merge upper bound sets.

Proposition 1. *(Ehrgott and Gandibleux, 2007) If U_1 and U_2 are upper bound sets for a same set $\bar{Y}$ and $U_1 - \mathbb{R}^p_{\geqq} \neq U_2 - \mathbb{R}^p_{\geqq}$ then $U^* := [(U_1 - \mathbb{R}^p_{\geqq}) \cap (U_2 - \mathbb{R}^p_{\geqq})]_N$ is an upper bound set for $\bar{Y}$ dominating U_1 and U_2.*

It is interesting to note that the simultaneous use of several upper bound sets can thus be a way to obtain a particularly tight upper bound set (Figure 2). We provide some additional explanations based on Figures 1 and 2. We consider here polyhedral upper bound sets U defined by a set of extreme points $\{y^1, \ldots, y^k\}$, *i.e.* $U = (\text{conv}\{y^1, \ldots, y^k\} - \mathbb{R}^2_{\geqq})_N$. $U_1 = (\text{conv } \{y_1, y_2, y_3\})_N$ and $U_2 = (\text{conv } \{y_4, y_5, y_6\})_N$ are incomparable bound sets, i.e. there is no dominance relation between them. The bound set $U^* := [(U_1 - \mathbb{R}^2_{\geqq}) \cap (U_2 - \mathbb{R}^2_{\geqq})]_N$ (which is $(\text{conv } \{y_4, y_7, y_3\})_N$) is obtained by merging U_1 and U_2. We can note that the point y_7 is not an extreme point of any of these later bound sets. This later point may not be feasible for the used relaxation, and even if it is, its importance is highlighted only when considering U_1 and U_2 simultaneously.

1.3. Surrogate Relaxation

The *surrogate relaxation* (Glover, 1975) is a well known relaxation that can be used to compute an upper bound on the optimal value of (*mDKP*) problem. Its principle is to aggregate all constraints of the problem using non-negative multipliers. In this subsection, we present some of its properties. Given a vector of multipliers $(u^1, \ldots, u^m) \in \mathbb{R}^m_{\geq}$, the surrogate relaxation is the

Figure 1: Example where U_1 and U_2 are two bound sets; they are incomparable

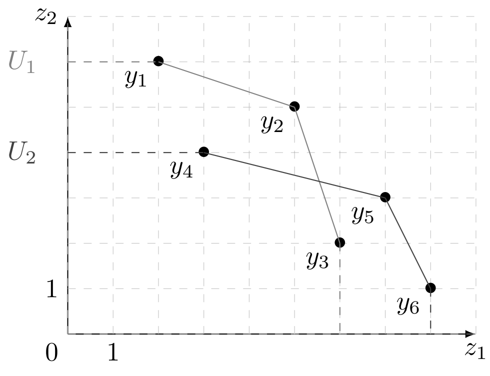
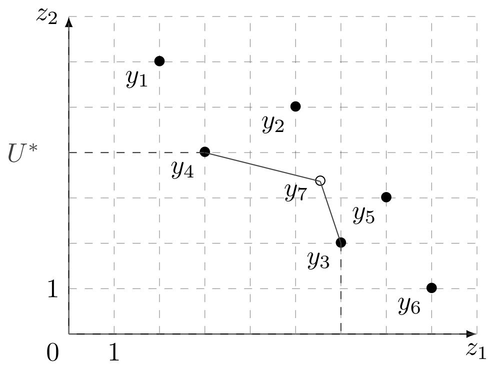


Figure 2: Bound set U^* obtained by merging U_1 and U_2



problem $SP(u_1,\ldots,u_m)$ defined by

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j\, x_j \\ &s.t. \sum_{i=1}^{m} u_i \sum_{j=1}^{n} w_{ij}\, x_j \le \sum_{i=1}^{m} u_i\, \omega_i \\ &\quad x_j \in \{0,1\} \qquad\qquad j=1,\ldots,n \qquad\qquad (SP(u_1,\ldots,u_m)) \end{aligned}$$

It is an instance of (KP) problem. Any solution method for this last problem can therefore be used to determine an optimal solution of $(SP(u_1,\ldots,u_m))$. Next, the associated optimal value defines an upper bound on the optimal value of the initial $(mDKP)$. However, the quality of the obtained upper bound depends on the used multiplier vector. The problem consisting in obtaining the best possible upper bound using a surrogate relaxation is called *dual surrogate problem DSP*:

$$\min_{(u_1,\ldots,u_m)\in\mathbb{R}^m_{\geqq}} \max \Big\{ \sum_{j=1}^{n} c_j\, x_j \mid \sum_{i=1}^{m} u_i \sum_{j=1}^{n} w_{ij}\, x_j \le \sum_{i=1}^{m} u_i\, \omega_i,\ x \in \{0,1\}^n \Big\} \qquad (DSP)$$

The dual surrogate problem is $\mathcal{NP}$-hard (Boyer, 2007). An exact solution method for the bi-dimensional case has been provided by Fréville and Plateau (1993). Several approximation methods for this problem have also been provided (see Wilbaut et al. (2008) for a recent survey, and Fleszar and Hindi (2009); Boyer et al. (2009); Puchinger et al. (2010) for recent contributions on the 0–1 multi-dimensional knapsack problem).

The surrogate relaxation can also be considered for the multi-objective case, we obtain thus an instance of $(pOKP)$.

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j^k\, x_j \qquad\qquad k=1,\ldots,p \\ &s.t. \sum_{i=1}^{m} u_i \sum_{j=1}^{n} w_{ij}\, x_j \le \sum_{i=1}^{m} u_i\, \omega_i \\ &\quad x_j \in \{0,1\} \qquad\qquad j=1,\ldots,n \qquad\qquad (pOSP(u_1,\ldots,u_m)) \end{aligned}$$

A distinction between feasible solutions and points of the initial problem $(pOmDKP)$ and its relaxation $(pOSP(u_1,\dots,u_m))$ will be necessary. We denote by $Y_N(u_1,\dots,u_m)$ the set of nondominated points of $(pOSP(u_1,\dots,u_m))$, and simply Y_N the set of nondominated points of $(pOmDKP)$. $Y_N(u_1,\dots,u_m)$ defines an upper bound set for Y_N. Obviously, the computation of this upper bound set is practically expensive. The set of nondominated extreme points $Y_{SN1}(u_1,\dots,u_m)$ of $(pOSP(u_1,\dots,u_m))$ defines an upper bound set $(\operatorname{conv} Y_{SN}(u_1,\dots,u_m))_N$ for $Y_N(u_1,\dots,u_m)$. By transitivity, it is thus also an upper bound set for Y_N. This last upper bound set has been used by da Silva et al. (2004) and Perederieiva (2011).

In the following, we will consider upper bound sets of the kind $(\operatorname{conv} Y_{SN}(u_1,u_2))_N$ for the specific bi-objective bi-dimensional case $(2O2DKP)$. This kind of upper bound set will be called *convex surrogate upper bound set* (CSUB). As multipliers take only non-negative values, we can divide both multipliers u_1 and u_2 by (u_1+u_2), so that their sum becomes equal to 1. In other words, only one multiplier u becomes necessary in order to specify a particular CSUB, that will be denoted by $CSUB(u) := (\operatorname{conv} Y_{SN}(u))_N$. The same way, surrogate relaxations will be denoted $(2OSP(u))$ and their sets of (extreme supported) efficient solutions and nondominated points will be denoted respectively $X_E(u)$ $(X_{SE1}(u))$ and $Y_N(u)$ $(Y_{SN1}(u))$.

The purpose of this paper is to determine the tightest possible bound set for Y_N using CSUB for the specific $(2O2DKP)$ problem. This upper bound set will be defined in Section 2.1. An attempt of adaptation of a classical dichomotic method will be proposed in Section 2.2, with some unexpected difficulties. In Section 3, theoretical foundations about the differentiation of CSUB depending on the used multiplier are provided. Based of this, an original approach enumerating all CSUB is proposed in Section 4. Afterwards, an improved algorithm based on dominance relations between CSUBs is proposed in Section 5, and a derived heuristic algorithm in Section 6. Numerical experiments are finally provided in Section 7.

2. Optimal convex surrogate upper bound set

2.1. Definition

The goal is to determine (efficiently) the tightest possible upper bound set that can be obtained for $(2O2DKP)$ using CSUBs. In other words, we aim to propose a generalization of the dual surrogate problem for the bi-objective case.

As stated in Section 1.2 two upper bound sets are not necessarily comparable, this is in particular true for CSUBs defined using different multipliers. Perederieiva (2011) has used Proposition 1 to merge a number h of incomparable CSUBs, and thus to obtain a tighter upper bound set.

Proposition 2. *(Perederieiva, 2011) ($\cap_{i=1}^{h}(CSUB(u^i) - \mathbb{R}^2_{\geqq}))_N$ is a valid upper bound set and it dominates any of the bound sets $(\operatorname{conv} Y_{SN}(u^i))_N$ used in the intersection.*

Obviously, considering a larger set of CSUBs in Proposition 2 allows to obtain a tighter upper bound set. Ideally, we would like to consider all possible CSUB. Since the number of feasible solutions of problems of the kind $(2OSP(u))$ is finite, then the number of possible CSUBs is also finite. Therefore, Proposition 2 can be applied to merge all CSUBs. We can thus give a first generalization of the dual surrogate problem.

Definition 3. $(\cap_{u\geq 0}(CSUB(u) - \mathbb{R}^2_{\geqq}))_N$ *is the tightest upper bound set based on the convex surrogate relaxation and the number of multipliers to consider in order to obtain it is finite. It is denoted the* optimal convex surrogate upper bound set *(OCSUB).*

Using the same arguments, another generalization of the dual surrogate problem is possible. Using Proposition 1 in order to merge bound sets of the kind $Y_N(u)$, we can obtain the tightest upper bound set based on the surrogate relaxation.

Definition 4. $(\cap_{u\geq 0}(Y_N(u) - \mathbb{R}^2_{\geqq}))_N$ *is the tightest upper bound set based on the surrogate relaxation and the number of multipliers to consider in order to obtain it is finite. It is denoted the* optimal surrogate upper bound set *(OSUB).*

The computation of the OSUB would be particularly expensive as it would require the solution of a number of $(2OKP)$ problems. This is why our aim is restricted the computation of the OCSUB.

2.2. Attempt for a dichotomic method

From Definition 3, we can deduce that the upper bound set OCSUB verifies the following properties: $(\text{OCSUB} - \mathbb{R}^2_{\geqq})$ is convex and $(\text{OCSUB} - \mathbb{R}^2_{\geqq})$ is a polyhedron. A first natural idea for the computation of the upper bound set OCSUB is to use the existing algorithms for the solution of the single-objective case. Indeed, the following result can be used.

Proposition 3. *(Ehrgott and Gandibleux, 2007) Consider a relaxation $(\tilde{P})$ of a MOCO problem (P), $(\tilde{P}_\lambda)$ is a relaxation of (P_λ) for all $\lambda \in \mathbb{R}^2_{\geq}$.*

Solving a surrogate relaxation (for any multiplier) of $(2O2DKP_\lambda)$ with $\lambda \in \mathbb{R}^2_{\geq}$, we obtain a solution $\tilde{x}$ the image of which in objective space is given by $\tilde{y} = z(\tilde{x})$. Proposition 3 implies that $Y_N \subset (\lambda^T \tilde{y} - \mathbb{R}^2_{\geqq})$. Using different directions $\lambda^1, \ldots, \lambda^k \in \mathbb{R}^2_{\geq}$ and denoting $y^1, \ldots, y^k$ the image of the optimal solutions of surrogate relaxations of the corresponding weighted sum problems, we obtain that

$$Y_N \subset \bigcap_{i=1}^{k} ({\lambda^i}^T y^i - \mathbb{R}^2_{\geqq}).$$

Thus $(\bigcap_{i=1}^k ({\lambda^i}^T \tilde{y}^i - \mathbb{R}^2_{\geqq}))_N$ is an upper bound set for Y_N. Obviously, this upper bound set will be tighter if we solve the dual surrogate problem for each considered weighted sum problem.

As $(\text{OCSUB} - \mathbb{R}^2_{\geqq})$ is a polyhedron, there exists a finite number of direction $\lambda^1, \ldots, \lambda^l$ defining the normal to each edge of OCSUB. Therefore we can compute OCSUB by solving exactly the dual surrogate problem associated to each of these weighted sum problems. However, we do not know these directions initially.

This problem seems similar to the computation of the convex relaxation $(\operatorname{conv} Y_{SN})_N$ of a bi-objective combinatorial optimization problem. Indeed $(\operatorname{conv} Y_{SN} - \mathbb{R}^2_{\geqq})$ is a polyhedron and its computation can be done by a dichotomic algorithm (Aneja and Nair, 1979). It seems thus very natural to apply the same algorithm with a minor modification: the exact solution of the weighted sum single-objective problem $(2DKP)$ would be replaced by the solution of the corresponding dual surrogate problem. Nevertheless, some difficulties will immediately appear.

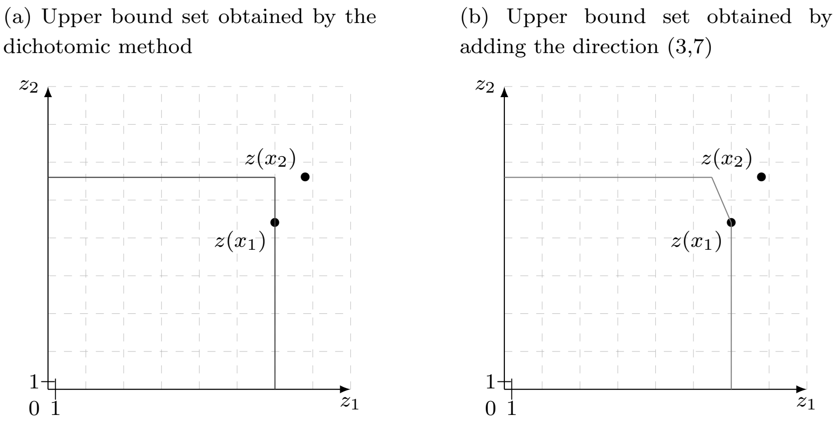
Example 1. *We consider the following instance of $(2O2DKP)$.*

$$\begin{array}{lrcrcrcrcrl}
\max & 10x_1 & + & 7x_2 & + & 20x_3 & + & 7x_4 & + & 8x_5 & \\
\max & 15x_1 & + & 17x_2 & + & 7x_3 & + & 4x_4 & + & 10x_5 & \\
s.t. & 3x_1 & + & 1x_2 & + & 9x_3 & + & 4x_4 & + & 9x_5 & \leq 13 \\
 & 13x_1 & + & 11x_2 & + & 2x_3 & + & 1x_4 & + & 7x_5 & \leq 17 \\
 & \multicolumn{10}{c}{x_j \in \{0,1\},\ j = 1\ldots 5}
\end{array} \qquad (2O2DKP\text{-}1)$$

We initialize the dichotomic method with the directions $\lambda^1 = (1,0)$ *and* $\lambda^2 = (0,1)$. *For* $\lambda^1 = (1,0)$, *the optimal solution obtained solving the dual surrogate problem (Fréville and Plateau, 1993) is* $x^1 = (1,0,1,0,0)$ *with* $z(x^1) = (30,22)$. *For* $\lambda^2 = (0,1)$, *the optimal solution obtained solving the dual surrogate problem is* $x^2 = (0,1,1,1,0)$ *with* $z(x^2) = (34,38)$. *We can immediately note that* x^2 *dominates* x^1. *Using these two points is thus inappropriate in order to define a direction to continue the application of the dichotomy. Thus, we can only conclude that* $Y_N \subset \{y \in \mathbb{R}^2 \ : \ y_1 \leq 30\} \cap \{y \in \mathbb{R}^2 \ : \ y_2 \leq 38\}$.

However by using the direction $\lambda^3 = (3,7)$, *we can improve the bound set (see Figure 3). Indeed, the optimal solution obtained solving the dual surrogate is then again* $x^1 = (1,0,1,0,0)$ *with* $z(x^1) = (30,22)$. *Finally, there is no immediate way to guess this direction using a dichotomic principle.*

Figure 3: Illustration on the example of the dichotomic method



The idea behind the attempt of dichotomic method is, knowing an appropriate direction defining an edge of the OCSUB, to deduce the associated multiplier. However, Example 1 shows that these directions are not straight-forward to find. In the following, we will consider another approach based on the computation of CSUBs.

3. CSUB and multiplier-set decomposition

Definition 3 shows that the computation of the OCSUB can be done by an enumeration of a finite set of CSUBs. A finite set of multipliers can therefore be used to compute these CSUBs. In this section, we study the association of multipliers to CSUBs. Ideally, we would like to obtain a multiplier-set decomposition with a one-to-one correspondence between subsets of multipliers and CSUBs. Such a decomposition would guarantee an exhaustive enumeration of all CSUB, without redundancy.

3.1. Critical multipliers and stability intervals

Differences between feasible sets of surrogate relaxations $(2OSP(u))$ for $u \in [0,1]$ are crucial in the differentiation of CSUBs. In this subsection, we consider individually solutions that are feasible for a problem $(2OSP(u))$ where $u \in [0,1]$. To shorten the notations, this kind of solution will be called u-surrogate feasible. An u-surrogate feasible solution x respects the constraint $u\,w_1(x) + (1-u)\,w_2(x) \leq u\,\omega_1 + (1-u)\,\omega_2$. Hence, two cases are possible: either x respects both constraints of $(2O2DKP)$ or x respects only one of them (if x does not respect any of the constraints, it does not respect this constraint neither).

In the first case, x is feasible for $(2O2DKP)$ and is thus u-surrogate feasible for any $u \in [0,1]$.

In the latter case, x is u-surrogate feasible for any $u \in [0,1]$ such that $u\,w_1(x)+(1-u)\,w_2(x) \leq u\,\omega_1 + (1-u)\,\omega_2$. This can be equivalently written $u\,(w_1(x) - w_2(x) - \omega_1 + \omega_2) \leq \omega_2 - w_2(x)$. Next, we have necessarily $w_1(x) - w_2(x) - \omega_1 + \omega_2 \neq 0$. Indeed, $w_1(x) - w_2(x) - \omega_1 + \omega_2 = 0$ would imply $w_1(x) - \omega_1 = w_2(x) - \omega_2$, *i.e.* either none or both constraints are satisfied (and we make the assumption that only one is satisfied). The constraint can thus be rewritten

$$\begin{cases} u \leq \frac{\omega_2 - w_2(x)}{w_1(x) - w_2(x) - \omega_1 + \omega_2} & \text{if } (w_1(x) - w_2(x) - \omega_1 + \omega_2) > 0, \\ u \geq \frac{\omega_2 - w_2(x)}{w_1(x) - w_2(x) - \omega_1 + \omega_2} & \text{if } (w_1(x) - w_2(x) - \omega_1 + \omega_2) < 0. \end{cases}$$

In other words, x is u-surrogate feasible for any

$$\begin{cases} u \in [0,v] & \text{if } (w_1(x) - w_2(x) - \omega_1 + \omega_2) > 0, \\ u \in [v,1] & \text{if } (w_1(x) - w_2(x) - \omega_1 + \omega_2) < 0, \end{cases}$$

where $v = \frac{\omega_2 - w_2(x)}{w_1(x) - w_2(x) - \omega_1 + \omega_2}$.

v is therefore a particular multiplier associated to x. Indeed, it defines one of the extremities of the interval of multipliers u such that x is u-surrogate feasible.

Definition 5. *(i) A* critical multiplier *u is a multiplier such that there exists at least one solution x that is u-surrogate feasible but not $u - \epsilon$-surrogate feasible or not $u + \epsilon$-surrogate feasible, for any $\epsilon > 0$. The solution x and the critical multiplier u are called* associated*.*

(ii) The stability interval *of a solution x is the interval of all multipliers u such that x is u-surrogate feasible.*

Finally, there are three kinds of stability interval:

- $[0,1]$ for a solution feasible for $(2O2DKP)$;
- $[0,v]$ for a solution feasible only for the second capacity constraint of $(2O2DKP)$. The critical multiplier v is said to be of *type 0M*;
- $[v,1]$ for a solution feasible only for the first capacity constraint of $(2O2DKP)$. The critical multiplier v is said to be of *type M1*.

As a critical multiplier can be associated to several solutions, it can be simultaneously 0M and M1.

3.2. Critical multipliers and CSUB

Two CSUBs are different if their sets of extreme points are not identical. It comes obviously from particular differences between the feasible sets of corresponding problems $(2OSP(u))$. Lemma 1 provides a necessary condition for two CSUBs to be different.

Lemma 1. *Let u_i and u_j be two different multipliers, if $CSUB(u_i)$ and $CSUB(u_j)$ are different then at least one of the extreme supported solutions of $(2OSP(u_i))$ and $(2OSP(u_j))$ is feasible for one problem and not for the other one.*

Proof. Let $x^1 \in X_{SE1}(u_i)$. Suppose that $z(x^1)$ is not an extreme point of $CSUB(u_j)$, then $x^1 \notin X_{SE1}(u_j)$. This is obviously the case if x^1 is not u_j-surrogate feasible (Figure 4a). Otherwise, two sub-cases need to be considered.

- There is an u_j-surrogate feasible solution x^2 dominating x^1 (Figure 4b). Therefore x^2 is not u_i-surrogate feasible as it would contradict the efficiency of x^1 for $(2OSP(u_i))$.
- There are two solutions $x^2, x^3 \in X_{SE1}(u_j)$ such that a point $y \in (z(x^2), z(x^3))$ weakly dominates $z(x^1)$ (Figure 4c). Therefore, x^2 and/or x^3 are not u_i-surrogate feasible as it would contradict the assumption that $x^1 \in X_{SE1}(u_i)$.

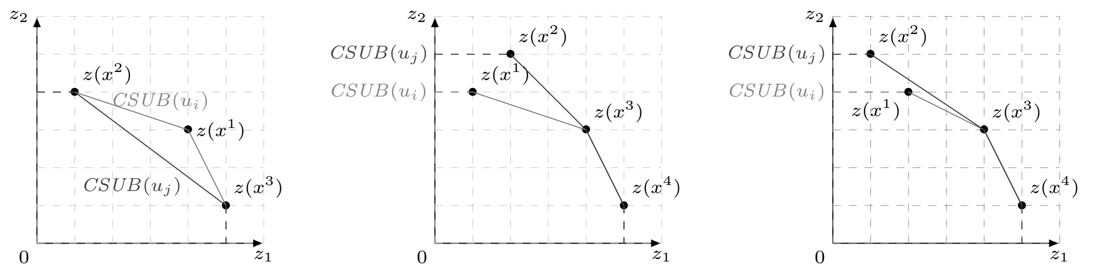
□

Figure 4: Illustration of the proof of Lemma 1

(a) x^1 u_i-surrogate feasible but not u_j-surrogate feasible

(b) x^2 dominates x^1

(c) x^1 is weakly dominated by a point of the segment $(z(x^2), z(x^3))$



Remark 1. *The converse of Lemma 1 is not true. Indeed, if the solution x^1 used in the proof of this Lemma is not u_j-surrogate feasible, there may exist another solution x^2 such that x^2 is u_j-surrogate feasible and $z(x^1) = z(x^2)$.*

Consequently to Lemma 1, the differences between CSUBs come only from the (in)feasibility of extreme supported solutions of corresponding surrogate relaxations. Thus, only critical multipliers associated to these solutions will be of interest.

Definition 6. *A critical multiplier associated to a solution $x \in X_{SE1}(u)$ is also called* critical multiplier associated to $CSUB(u)$, *and x is called* solution associated to $CSUB(u)$.

Given two multipliers u and u', the proof of Lemma 1 provides an analysis of the reason why a solution x associated to $CSUB(u)$, is not necessarily associated to $CSUB(u')$: either this solution is not u'-surrogate feasible or $x \notin X_{SE1}(u')$.

Definition 7. *Let x be a solution associated to a CSUB, if x is u-surrogate feasible and $x \notin X_{SE1}(u)$ then x is called* u-masked. *More precisely, if there is a u-surrogate feasible solution x' such that x' dominates x, we will say that x is u-masked by x'. If there are two u-surrogate feasible solutions x' and x'' such that $z(x)$ is weakly dominated by a point in $(z(x'), z(x''))$, we will say that x is u-masked by the pair of solutions x' and x''. More generally, a solution that is u-masked for all u in a set U will be called* masked on U.

Knowing the list of all multipliers associated to all possible CSUB, Proposition 4 allows to find a finite set of multipliers in order to compute the set of all CSUBs.

Proposition 4. *Suppose we know the list of all multipliers associated to all CSUBs, then the list of all CSUBs is given by the following enumeration:*

(i) $CSUB(0)$ *and* $CSUB(1)$*;*

(ii) *For all interval defined by two consecutive critical multipliers :* $CSUB(u)$ *for a multiplier* u *in the relative interior of this interval;*

(iii) $CSUB(u)$ *for all multiplier* u *that is simultaneously 0M and M1.*

Proof. Lemma 1 implies that $CSUB(u)$ is identical for all $u \in (v_1, v_2)$ where v_1 and v_2 are consecutive critical multipliers. It is thus useless to consider two multipliers in such an open interval (v_1, v_2). On the contrary, $CSUB(u)$ and $CSUB(u')$ such that u and u' belong to the interior of different intervals given by consecutive critical multipliers, implies that $CSUB(u)$ and $CSUB(u')$ may be different (but are not necessarily as the converse of Lemma 1 is not true). Knowing $CSUB(u)$ for u multiplier in the interior of all interval defined by consecutive critical multipliers is thus necessary to guarantee an exhaustive enumeration of all CSUBs (item *(ii)*). If 0 and 1 are not critical multiplier, one multiplier must also be considered in the interval $[0, u_{\min})$ and $(u_{\max}, 1]$ where $u_{\min}$ and $u_{\max}$ are respectively the smallest and the greatest critical multiplier, $CSUB(0)$ and $CSUB(1)$ are appropriate choices. Nevertheless, it is not sufficient to obtain an exhaustive enumeration of CSUBs. Indeed if we consider a multiplier v that is simultaneously 0M and M1, and a small number $\epsilon > 0$ such that there is no other critical multiplier in $[v - \epsilon, v + \epsilon]$. As v is 0M then $CSUB(v)$ may be different to $CSUB(v + \epsilon)$, and as v is M1 then $CSUB(v)$ may be different to $CSUB(v - \epsilon)$. The computation of $CSUB(v)$ is thus also necessary (item *(iii)*). For the same reason, the computation of $CSUB(0)$ (respectively $CSUB(1)$) is also required if 0 is a 0M multiplier (respectively 1 is a M1 multiplier). Computing $CSUB(0)$ and $CSUB(1)$ is therefore required in all cases (item *(i)*). □

The multiplier set-decomposition described by Proposition 4 does not exactly give a one-to-one correspondence between subsets of multipliers and CSUBs (as the converse of Lemma 1 is not true) but guarantees the determination of all CSUB. However, a way to obtain the set of all critical multipliers is necessary in order to use this last statement. Proposition 5 is based on an assumption that is easier to verify: only a subset of CSUBs with their associated critical multipliers are known. In this context, there may exist unknown solutions associated to unknown CSUBs, that are either u-masked or not u-surrogate feasible for each multiplier u considered so far. In order to determine these unknown solutions, we must find multipliers u such that these solutions are potentially u-surrogate feasible and not u-masked. A characterization of the intervals of multipliers S on which a solution is masked is provided by Proposition 5.

Proposition 5. *Suppose we know some CSUBs, with their associated solutions and critical multipliers. Let* x *be an unknown solution associated to a CSUB, then* x *is* u*-masked for all* $u \in U$ *where* U *is an union of intervals. The possible patterns for these intervals, relies on the stability interval of* x*.*

(1) If the stability interval of x *is* $[0, 1]$*, then these patterns are:*

(i) $[0, v_0]$ *where v_0 is a known 0M multiplier;*

(ii) $[v_1, 1]$ *where v_1 is a M1 multiplier ;*

(iii) $[v_1, v_0]$ *where v_0 and v_1 are respectively known 0M and M1 multipliers.*

(2) If the stability interval of x is $[0, v]$, then these patterns are:

(i) $[0, v_0]$ *where v_0 is a known 0M multiplier such that $v_0 < v$;*

(ii) $[v_1, v]$ *where v_1 is a known 0M multiplier such that $v_1 \leq v$;*

(iii) $[v_1, v_0]$ *where v_0 (such that $v_0 \leq v$) and v_1 are respectively known 0M and M1 multipliers.*

(3) If the stability interval of x is $[v, 1]$:

(i) $[v, v_0]$ *where v_0 is a known 0M multiplier such that $v \leq v_0$;*

(ii) $[v_1, 1]$ *where v_1 is a known M1 multiplier such that $v < v_1$;*

(iii) $[v_1, v_0]$ *where v_0 and v_1 (such that $v \leq v_1$) are respectively a 0M and M1 critical multipliers.*

Proof. (1) We assume that the stability interval of x is $[0, 1]$.

We suppose first that x is u'-masked by one solution x' for a given multiplier u', then x is u-masked for all multiplier u such that x' is u-surrogate feasible. In other words, if we denote by S the stability interval of x', then x is masked on S. The interval S is either of the kind $[0, v_0]$ (pattern (i)) or $[v_1, 1]$ (pattern (ii)) where v_0 and v_1 are respectively 0M and M1 multipliers. The case $S = [0, 1]$ is not possible since x is a solution associated to a CSUB.

We suppose next that x is u'-masked by the pair of solutions x' and x'' for a given multiplier u', then x is u-masked for all multipliers u such that both solutions are u-surrogate feasible. In other words, if we denote the stability interval of both solutions by S_1 and S_2 then x is masked on $S_1 \cap S_2$. If both solutions are associated to 0M critical multipliers v_{01}, v_{02} then x is masked on $[0, \min(v_{01}, v_{02})]$ (pattern (i)). If both solutions are associated to M1 critical multipliers v_{11}, v_{12}, then x is masked on $[\max(v_{11}, v_{12}), 1]$ (pattern (ii)). If one solution is associated to a 0M critical multiplier v_0 and the other one is associated to a M1 critical multiplier v_1, then x is masked on $[v_1, v_0]$ (pattern (iii)).

(2) We assume that the stability interval of x is $[0, v]$ where u is a 0M critical multiplier. In order to find the possible pattern for this second case, we just need to compute the intersection of $[0, v]$ with the pattern enumerated in the case (1). For the pattern (1)(i), we necessarily have $[0, v_0] \cap [0, v] = [0, v_0]$ with $v > v_0$ (pattern (i)), otherwise x is u-masked for all u in its stability interval. For the pattern (1)(ii), we either have $[v_1, 1] \cap [0, v] = [v_1, v]$ with $v \geq v_1$ (pattern (ii)) or $[v_1, 1] \cap [0, v] = \emptyset$ if $v < v_1$. For the pattern (iii), we have either $[v_1, v_0] \cap [0, v] = [v_1, v_0]$ (pattern (iii)) with $v_1 < v_0 \leq v$, or $[v_1, v_0] \cap [0, v] = [v_1, v]$ (pattern (ii)) with $v_1 \leq v < v_0$ (we always have $v_1 \leq v$ since solutions associated to v_0 and v_1 are a pair masking x).

(3) The proof is symmetric to the case (2).

□

Remark 2. *Given a unknown solution x associated to a CSUB, the (disjoint) union of intervals U on which x is masked by known solutions, is composed of zero or one interval of type $[0, v_0]$ (or $[v, v_0]$ for the case (3)), zero or one interval of type $[v_1, 1]$ (or $[v, v_0]$ for the case (2)) and zero or several disjoint intervals of type $[v_1, v_0]$.*

The exact computation of the union of intervals U would require to know the solution x. Nevertheless, Proposition 5 only gives the patterns of intervals on which an unknown x is masked. However, as the boundaries of these intervals (except v) are defined by known multipliers, it will be possible to deduce particular multipliers u such that potential unknown solutions associated to CSUBs are not u-masked by known solutions or pairs of solutions.

4. TotalEnumerative algorithm

In this section, we describe a first enumerative algorithm for the computation of the OCSUB. The main idea of this algorithm is to determine first the set of all multipliers associated to all possible CSUB (line 2 of Algorithm 1), and next to deduce the set of all CSUBs (line 3 of Algorithm 1). The OCSUB can finally be computed by an application of Definition 3.

Algorithm 1: Algorithm `TotalEnumerative`

```
  input  : A $2O2DKP$ Instance
  output: bound: the OCSUB for the instance of $2O2DKP$
1 begin
      /* allMult denotes the set of all enumerated multipliers      */
      /* bound denotes the bound obtained by merging CSUBs          */
2     EnumerateCriticalMultiplier(allMult $\uparrow$, bound $\uparrow$)
3     ComputeOCSUB(allMult $\downarrow$, bound $\updownarrow$)
```

Comment. In the algorithms, the symbols $\downarrow$, $\uparrow$ and $\updownarrow$ specify the transmission mode of a parameter to a procedure; they correspond respectively to the mode IN, OUT and IN OUT.

4.1. First part: the enumeration of all critical multipliers associated to all possible CSUB

In order to initialize our enumeration, we start by computing $CSUB(0)$ and $CSUB(1)$ and deducing their associated multipliers (line 2 of Algorithm 2). The computation of these CSUBs is indeed required in all cases according to Proposition 4. Another interesting property about these two multipliers is given by Lemma 2.

Lemma 2. *All solutions associated to any CSUB are either 0-surrogate feasible or 1-surrogate feasible or both.*

Proof. We already know that the stability interval of solutions associated to CSUBs are of three kinds: $[0, u]$, $[u, 1]$, $[0, 1]$. The statement follows. □

Consequently to this lemma, the union of intervals on which an unknown solution x (associated to an unknown CSUB) is masked by a known solution cannot be empty after this initialization.

According to Lemma 1, a way to enumerate new CSUBs is to consider multipliers u such that at least one solution associated to a known CSUB is no longer u-surrogate feasible. In other words, we need to consider multipliers u outside of the stability interval of known solutions. There will be two cases to consider, a solution is associated to a 0M multiplier or a M1 multiplier (there is nothing to do in the case of a solution the stability interval of which is $[0, 1]$ as it is u-surrogate feasible for all $u \in [0, 1]$). In the case of a 0M multiplier v_0, any multiplier $u > v_0$ will guarantee that the solution(s) associated to v_0 is (are) no longer u-surrogate feasible. Symmetrically, any multiplier $u < v_1$ will guarantee that the solution(s) associated to a M1 multiplier v_1 is (are) no longer u-surrogate feasible.

In an iteration of the procedure (line 5-10 of Algorithm 2), a critical multiplier v is considered individually (a multiplier simultaneously 0M and M1 is here considered as two different multipliers) to define a multiplier u the value of which is as near as possible to v. Ideally, there must not be any critical multiplier between v and u. Thus, appropriate values u are defined by $u := v+\epsilon$ if u is a 0M multiplier, and $u := v-\epsilon$ if u is a M1 multiplier, where $\epsilon > 0$ is a small value (see appendix Appendix A for an upper bound on ϵ). $CSUB(u)$ is next computed; and its associated multipliers are deduced.

All CSUBs computed during the execution of this algorithm are merged using Proposition 2 (line 4,10 of Algorithm 2).

Algorithm 2: Procedure `EnumerateCriticalMultiplier`

```
input  : Instance of $(2O2DKP)$
output: allMult: the list of all critical multipliers associated to all possible CSUB
bound: An upper bound set for $Y_N$
1  begin
       /* compAllCM computes all critical multipliers associated to a CSUB           */
       /* multToDo denotes the multipliers from which deriving a new multiplier      */
       /* Here, 0M and M1 multiplier is considered as two distinct multipliers       */
2      allMult ← compAllCM($CSUB(0)\downarrow$) ∪ compAllCM($CSUB(1)\downarrow$)
3      multToDo ← allMult
4      bound ← $[(CSUB(0) - \mathbb{R}^2_{\geqq}) \cap (CSUB(1) - \mathbb{R}^2_{\geqq})]_N$
5      while multToDo ≠ ∅ do
           /* Select a multiplier into the list                                      */
6          $v$ ← getMult(multToDo ↓); multToDo ← multToDo \ {$v$}
7          if $v$ is 0M then $u \leftarrow v+\epsilon$ else $u \leftarrow v-\epsilon$
8          multToDo ← multToDo ∪ [compAllCM($CSUB(u)\downarrow$) \ allMult]
9          allMult ← allMult ∪ compAllCM($CSUB(u)\downarrow$)
10         bound ← $[(\text{bound} - \mathbb{R}^2_{\geqq}) \cap (CSUB(u) - \mathbb{R}^2_{\geqq})]_N$
```

Proposition 6. *Algorithm 2 finds all critical multipliers associated to all possible CSUB.*

Proof. Let u be a critical multiplier associated to a CSUB. We denote by x a solution associated to u. If u is associated to $CSUB(0)$ or $CSUB(1)$, then u is found during the initialization. If u is not associated to either $CSUB(0)$ nor $CSUB(1)$, then we show that v is found in an iteration of Algorithm 2. We consider thus an iteration of this algorithm and suppose that v has not been found in a preceding iteration. Proposition 5 states the pattern of intervals U the union on which x is masked by known solutions. According to Lemma 2, there is at least one interval U defined by Proposition 5 on which x is masked. The interval U has either a left-extremity defined by a M1 multiplier v_1 or a right-extremity defined by a 0M multiplier v_0 or both. Algorithm 2 considers the computation of all CSUBs of the following kind: $CSUB(v_0 + \epsilon)$, $CSUB(v_1 - \epsilon)$, with possible new solutions and critical multipliers found. There are two possible conclusions: u is one of these new critical multipliers or not (*i.e.* all solutions associated to u are still masked). In the latter case, new critical multipliers allow to performs new iterations. Again, u will be found or not. By repeated computations of CSUBs, u will be finally found in a finite number of computations of CSUB, as the number of critical multipliers is finite and as multipliers v such that x is not v-masked by known solution(s) can always be deduced.

□

4.2. Second part: computation of the OCSUB

After the first part, all critical multipliers associated to all CSUBs are known. Thus, Proposition 4 can be used in order to compute all CSUBs. We summarize the CSUBs computed in the part 1:

(i) $CSUB(0)$ and $CSUB(1)$ in the initialization;

(ii) $CSUB(u+\epsilon)$ for all u 0M multiplier;

(iii) $CSUB(u-\epsilon)$ for all u M1 multiplier.

Items (*ii*) and (*iii*) of the enumeration above imply that at least one multiplier has been used in the interior of intervals defined by consecutive critical multipliers such that the left hand side is 0M or the right hand side is M1. To complete Item (*ii*) of Proposition 4, it remains thus to use one multiplier in the interval the left side of which is M1 and the right side of which is 0M. If we denote by u the left hand side of this last kind of interval, the multiplier $u+\epsilon$ is an appropriate choice for the computation of a CSUB.

In conclusion, to achieve the enumeration of all CSUBs, it remains to compute:

- $CSUB(u+\epsilon)$ for all interval of consecutive critical multipliers $[u, u_0]$ such that u is M1 and u_0 is 0M (Item (*ii*) of Proposition 4);
- $CSUB(u)$ for all multiplier u that is 0M and M1 (Item (*iii*) of Proposition 4).

Algorithm 3 summarizes this final computation.

Algorithm 3: Procedure `ComputeOCSUB`

```
input  : A $2O2DKP$ instance, the set of all critical multipliers allMult,
bound: an upper bound set bound for $Y_N$
output: bound: the OCSUB for the instance of $2O2DKP$
1 begin
2     foreach $u \in$ allMult do
3         if $u$ is 0M and M1 then bound $\leftarrow [(\text{bound} - \mathbb{R}^2_{\geqq}) \cap (CSUB(u) - \mathbb{R}^2_{\geqq})]_N$
4     foreach Interval of consecutive critical multipliers $[u, v]$ such that $u$ is M1 and $v$ is 0M do
5         bound $\leftarrow [(\text{bound} - \mathbb{R}^2_{\geqq}) \cap (CSUB(u+\epsilon) - \mathbb{R}^2_{\geqq})]_N$
```

Example 1 (continued).

The *TotalEnumerative* algorithm is performed on the instance of ($2O2DKP$) refered as Example 1. To facilitate the understanding of the output, Table 1 reports all solutions associated to at least one CSUB and their stability interval.

The solutions are represented in objective space on Figure 5a. The stability intervals of the solutions are represented on Figure 5b. On the representation of the stability interval of a solution x_i, the dashed parts correspond to the interval of multipliers on which x_i is masked.

The execution of the algorithm is presented in Table 2. The OCSUB is defined by the set of extreme points (rounded to 10^{-2}) {(30,22); (27.43,28)}. In checking Figure 5a and Table 2, a number of obtained CSUBs, outcomed of the TotalEnumerative algorithm, appears dominated by other CSUBs. Such a situation may appear in general, and these dominated CSUBs are therefore useless for the computation of the OCSUB. The next section gives an analysis of the dominance relation between the different CSUBs.

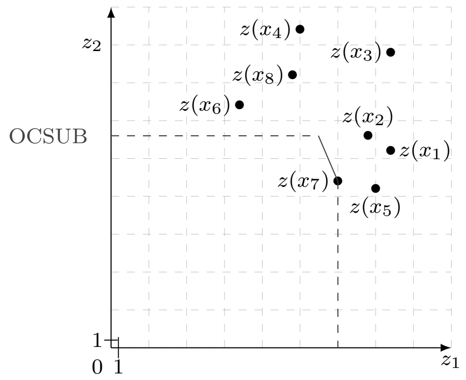
5. 0M-M1 interval algorithm

5.1. Bound sets and dominance

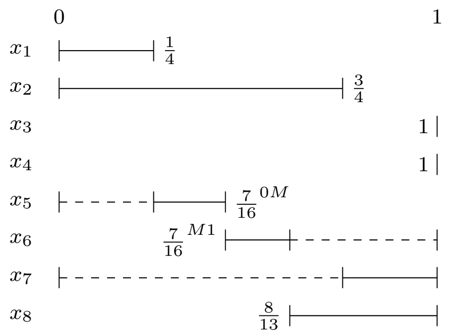
The dominance between CSUBs is based on the following Lemma, illustrated by Figure 4 (in Section 3.1 Page 10).

Table 1: Information relative to the solutions associated to at least one CSUB

Name	Solution	Objective value	Stability interval
x_1	(1,0,1,1,0)	(37,26)	$[0, \frac{1}{4}]$
x_2	(0,1,1,1,0)	(34,28)	$[0, \frac{3}{4}]$
x_3	(1,1,1,0,0)	(37,39)	$[1, 1]$
x_4	(1,1,0,0,1)	(25,42)	$[1, 1]$
x_5	(0,0,1,1,1)	(35,21)	$[0, \frac{7}{16}]$
x_6	(1,1,0,0,0)	(17,32)	$[\frac{7}{16}, 1]$
x_7	(1,0,1,0,0)	(30,22)	$[0, 1]$
x_8	(1,1,0,1,0)	(24,36)	$[\frac{8}{13}, 1]$



(a) The solutions in the objective space



(b) Stability interval of the solutions

Figure 5: Information relative to the eight solutions associated to a CSUB

Lemma 3. *We consider $CSUB(u_i)$ and $CSUB(u_j)$ two different CSUBs, defined respectively by the set of extreme points P^i and P^j :*

1. *If $P^j \subset P^i$ then $CSUB(u_j)$ dominates $CSUB(u_i)$ (Figure 4a)*
2. *If $P^j = P^{j_1} \cup P^{j_2}$ such that $P^{j_1} \subset P^i$ and the points of P^{j_2} are either dominated by points of P^i (Figure 4b) or weakly dominated by interior points in the edges defined by two points of P^i (Figure 4c), then $CSUB(u_j)$ dominates $CSUB(u_i)$.*

Dominated CSUBs do not contribute to the computation of the OCSUB. It is therefore useless to determine such CSUBs. However, it is difficult to know that a CSUB is dominated by another one before its computation. In the following, we provide a characterization allowing to know without additional computation that some CSUBs are dominated.

Given a known critical multiplier u, we consider $CSUB(u - \epsilon)$ and $CSUB(u + \epsilon)$ with ϵ small enough such that u is the only critical multiplier associated to a CSUB in the interval $[u - \epsilon, u + \epsilon]$. Suppose u is 0M and not M1, then there exists at least one solution associated to $CSUB(u - \epsilon)$ which is not $u + \epsilon$-surrogate feasible. Since u is not M1, there does not exist a solution associated to $CSUB(u+\epsilon)$ that is not $u-\epsilon$-surrogate feasible. Thus, by application of Lemma 3, either $CSUB(u+\epsilon)$ dominates $CSUB(u-\epsilon)$, or $CSUB(u+\epsilon) = CSUB(u-\epsilon)$ (case of equivalent solutions). In both cases, the computation of $CSUB(u-\epsilon)$ is useless. We will say that $CSUB(u + \epsilon)$ *weakly dominates* $CSUB(u - \epsilon)$.

We can do the symmetric analysis in the case u is M1 but not 0M and a similar one if u is both 0M and M1. Proposition 7 summarizes this result.

Table 2: Example of execution of Algorithm 1. NB: the value of $u = \frac{7}{16}$ corresponds simultaneously to one 0M and one M1. To avoid any confusion, the type of origin which is derived the value is mentioned in superscript.

	Nb CSUB	u	type	e	$CSUB(u+e)$	multToDo	multipliers
Initialization	1	0	-	0	$\{x_1, x_2\}$	$\mathbf{\frac{1}{4}}, \mathbf{\frac{3}{4}}$	$\mathbf{\frac{1}{4}}, \mathbf{\frac{3}{4}}$
	2	1	-	0	$\{x_3, x_4\}$	$\frac{1}{4}, \frac{3}{4}, \mathbf{1}$	$\frac{1}{4}, \frac{3}{4}, \mathbf{1}$
First part	3	$\frac{1}{4}$	0M	$+\epsilon$	$\{x_2, x_5\}$	$\mathbf{\frac{7}{16}}^{\mathbf{0M}}, \frac{3}{4}, 1$	$\frac{1}{4}, \mathbf{\frac{7}{16}}, \frac{3}{4}, 1$
	4	$\frac{7}{16}^{0M}$	0M	$+\epsilon$	$\{x_2, x_6\}$	$\mathbf{\frac{7}{16}}^{\mathbf{M1}}, \frac{3}{4}, 1$	$\frac{1}{4}, \mathbf{\frac{7}{16}}, \frac{3}{4}, 1$
	5	$\frac{7}{16}^{M1}$	M1	$-\epsilon$	$\{x_2, x_5\}$	$\frac{3}{4}, 1$	$\frac{1}{4}, \frac{7}{16}, \frac{3}{4}, 1$
	6	$\frac{3}{4}$	0M	$+\epsilon$	$\{x_7, x_8\}$	$\mathbf{\frac{8}{13}}, 1$	$\frac{1}{4}, \frac{7}{16}, \mathbf{\frac{8}{13}}, \frac{3}{4}, 1$
	7	$\frac{8}{13}$	M1	$-\epsilon$	$\{x_2, x_6\}$	1	$\frac{1}{4}, \frac{7}{16}, \frac{8}{13}, \frac{3}{4}, 1$
	8	1	M1	$-\epsilon$	$\{x_7, x_8\}$	$\emptyset$	$\frac{1}{4}, \frac{7}{16}, \frac{8}{13}, \frac{3}{4}, 1$
Second part	9	$\frac{7}{16}$	0M&M1	0	$\{x_2, x_5, x_6\}$	-	"
	10	$\frac{8}{13}$	M1	$+\epsilon$	$\{x_2, x_8\}$	-	"

Proposition 7. *1. If a multiplier u is 0M and not M1 then $CSUB(u+\epsilon)$ weakly dominates $CSUB(u-\epsilon) = CSUB(u)$.*

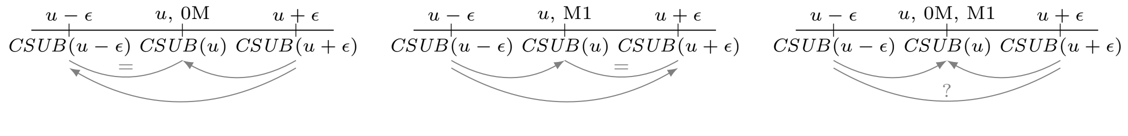
2. If a multiplier u is M1 but not 0M then $CSUB(u-\epsilon)$ weakly dominates $CSUB(u+\epsilon) = CSUB(u)$.

3. If a multiplier u is 0M and M1 then $CSUB(u-\epsilon)$ and $CSUB(u+\epsilon)$ may not be comparable. Both weakly dominate $CSUB(u)$.

If 0 is a critical multiplier, then 0 is 0M and Proposition 7 holds, except that we ignore the comparison with $CSUB(0-\epsilon)$ which is not defined. Similarly if 1 is a critical multiplier, it is M1 and $CSUB(1+\epsilon)$ is not defined.

Proposition 7 is illustrated in Figure 6, an arrow from $CSUB(u_j)$ to $CSUB(u_i)$ means that $CSUB(u_j)$ weakly dominates $CSUB(u_i)$.

Figure 6: Properties of 0M and M1 multipliers



Proposition 7 allows only a local analysis around critical multipliers. We will say that we can know *a priori* that a CSUB is weakly dominated. In the following, we apply the result of this proposition on the set of all critical multipliers associated to any CSUB, to obtain a stronger result.

Definition 8. *Let L be a list of critical multipliers sorted by increasing values. We call* 0M-M1 interval *in L any pair $u_0 - u_1$ of successive critical multipliers in $L \cup \{0, 1\}$, such that u_0 is 0M and u_1 is M1 and $u_0 < u_1$. 0 and 1 are respectively considered as 0M and M1 multipliers.*

Proposition 8. *We denote by L_{all} the list of all critical multipliers associated to all CSUBs. The only CSUBs that are not weakly dominated* a priori *are $CSUB(u)$ where $u_0 < u < u_1$ for any 0M-M1 interval $u_0 - u_1$ in L_{all}.*

Proof. We consider all possible types for a pair of critical multipliers u_1 and u_2 and we analyze the dominance relation between the CSUBs around these two critical multipliers. We only present the case for which u_1 is 0M (but not M1). The proof is similar if it is M1 or, 0M and M1. As u_1 is 0M, Proposition 7 implies that $CSUB(u_1+\epsilon)$ weakly dominates $CSUB(u_1) = CSUB(u_1-\epsilon)$. As there is no critical multiplier between u_1 and u_2, $CSUB(u_1+\epsilon) = CSUB(u_2-\epsilon)$. The question is next whether $CSUB(u_1+\epsilon)$ is weakly dominated or not by $CSUB(u_2+\epsilon)$ or $CSUB(u_2)$. In all cases, the answer comes from the application Proposition 7 for the multiplier u_2.

1. If u_2 is 0M and not M1, then $CSUB(u_1+\epsilon)$ is weakly dominated by $CSUB(u_2+\epsilon)$.
2. If u_2 is M1 and not 0M, then $CSUB(u_1+\epsilon)$ weakly dominates $CSUB(u_2+\epsilon)$.
3. If u_2 is 0M and M1, then $CSUB(u_1+\epsilon)$ weakly dominates $CSUB(u_2)$.

The only cases for which $CSUB(u_1+\epsilon) = CSUB(u_2-\epsilon)$ is not dominated a priori, happen when u_2 is M1.

Suppose 0 is not a critical multiplier, and we denote the smallest critical multiplier by u_{min}. The same CSUB is therefore obtained for $CSUB(u)$ for $u \in [0, u_{min})$. Depending of the type of u_{min}, $CSUB(u_{min}-\epsilon)$ is weakly dominated ($u_{\min}$ is 0M) or not ($u_{\min}$ is M1). The case of the multiplier 1 is symmetric.

□

Figure 7: Illustration of Proposition 8

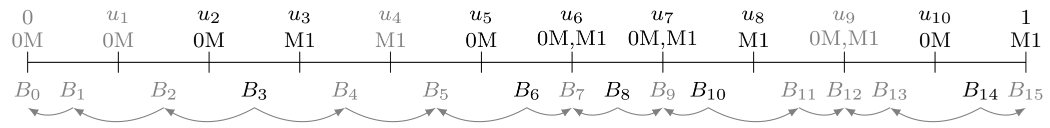


Figure 7 illustrates Propositions 7 and 8. Dominance between CSUBs is again represented by an arrow, and the CSUBs that are not dominated a priori are denoted in solid black.

5.2. 0M-M1 interval algorithm

The *0M-M1 interval* algorithm is based on Proposition 8 which states that the only multiplier u such that $CSUB(u)$ is not dominated *a priori* are in the interior of the 0M-M1 intervals in L_{all}. Obviously, the direct use of this result requires to know L_{all}. Knowing only a subset of CSUBs with their associated solutions and critical multipliers, Proposition 5 defines an union of (patterns of) intervals U such that an unknown solution associated to a critical multiplier v is masked on U. To find such an unknown solution x, it is necessary to compute $CSUB(u)$ for $u \in S \setminus U$ where S is the stability interval of x. According to Proposition 5, U is composed of zero or one interval of type $[0, v_0]$ (or $[v, v_0]$ if v is a M1 multiplier), zero or one interval of type $[v_1, 1]$ (or $[v_1, v]$ if v is a 0M multiplier) and zero or several disjoint intervals of type $[v_1, v_0]$ where the multipliers denoted v_0 and v_1 are known 0M and M1 multipliers. It is interesting to note that the set $S \setminus U$ is an union of 0M-M1 intervals defined by a subset of known multipliers, with the exception of at most one interval for which the extremities are a known critical multiplier and v (unknown).

The 0M-M1 algorithm (algorithm 4) is based on these observations. An iteration of the algorithm is performed as follows: given a 0M-M1 interval $u_0 - u_1$ in the list of known multipliers, $CSUB(u_0+\epsilon)$ is computed and the new critical multipliers are stored. We call this operation *the exploration of the 0M-M1 interval.* Since the exploration of a 0M-M1 interval is based on the 0M multiplier, we store the 0M multiplier for each 0M-M1 interval explored, in order to avoid repetitive computations of a same CSUB. Indeed, new multipliers may be found with the computation of a new CSUB. If a new M1 multiplier is inserted between a pair of 0M and M1 multiplier, a new 0M-M1 interval is defined, but its exploration is done by computing the same CSUB.

The initialization is reduced to the computation of $CSUB(0)$. Indeed $CSUB(1)$ is either dominated or redundant with the exploration of a 0M-M1 interval.

Algorithm 4: 0M-M1 interval algorithm

```
input  : A 2O2DKP instance
output: The OCSUB of the instance of 2O2DKP
1 begin
      /* multipliers is sorted in non increasing order, 1 is considered as a M1 multiplier
         */
2     bound ← $CSUB(0)$
3     multipliers ← compAllCM($CSUB(0)\downarrow$) ∪ {1}
4     0MDone ← ∅
5     while there exists a 0M-M1 interval $u_0 - u_1$ such that $u_0 \notin$ 0MDone do
6         0MDone ← 0MDone ∪ {$u_0$}
7         bound ← $[(\text{bound} - \mathbb{R}^2_{\geqq}) \cap (CSUB(u_0+\epsilon) - \mathbb{R}^2_{\geq})]_N$
8         multipliers ← multipliers ∪ compAllCM($CSUB(u_0+\epsilon)\downarrow$)
9     return bound
```

5.3. *Correctness of the* 0M-M1 interval algorithm

In this part we prove the correctness of the *0M-M1 interval algorithm.* In this proof, we denote by L_{alg} the list of known critical multipliers at the end of Algorithm 4. This proof is decomposed in two propositions, its main idea is to show that the list L_{alg} contains at least the minimum number of critical multipliers to guarantee that the OCSUB is obtained as an output of the algorithm. As Algorithm 4 computes $CSUB(u_0+\epsilon)$ for each 0M-M1 interval $u_0 - u_1$ in L_{alg}, it is necessary for L_{alg} to contain all 0M multiplier that defines a 0M-M1 in the list L_{all} (Proposition 9). Moreover to ensure that $CSUB(u_0+\epsilon)$ is computed for all u_0 defining a 0M-M1 interval $u_0 - u_1$ in L_{all}, L_{alg} must also contain a M1 multiplier $u_2 > u_0$ such that there does not exist any 0M multiplier in (u_0, u_2) (Proposition 10).

We can note that for a same u_0 defining a 0M-M1 interval $u_0 - u_1$ in L_{all}, several M1 critical multipliers are appropriate to ensure the computation of $CSUB(u_0+\epsilon)$. To illustrate this, we consider the example of Figure 7. The CSUBs to be computed to guarantee that the OCSUB is found, are $CSUB(u+\epsilon)$ with $u = u_2, u_5, u_6, u_7, u_{10}$. L_{alg} must therefore contain these 0M critical multipliers. The M1 critical multiplier that may cause the exploration of the 0M-M1 interval $u_2 - u_3$ (in L_{all}) can be either u_3 or u_4. Indeed there does not exist any 0M critical multiplier in (u_2, u_3) or in (u_2, u_4), thus the explorations of $u_2 - u_3$ and $u_2 - u_4$ are the same: $CSUB(u_2+\epsilon)$. The other M1 multipliers to find are u_6, u_7 and, u_8 or u_9.

Proposition 9. *Algorithm 4 finds all multiplier u_0 such that there is a 0M-M1 interval $u_0 - u_1$ in L_{all}.*

Proof. Let $u_0 - u_1$ be a 0M-M1 interval in L_{all}, and x_0 be a solution associated to u_0. We show in the following that x_0 (and thus u_0) is found by Algorithm 4. If x_0 is associated to $CSUB(0)$, then x_0 is found with the initialization. If x_0 is not associated to $CSUB(0)$ then it is 0-masked (since x_0 is 0-surrogate feasible). By Proposition 5, the union of intervals on which x_0 is masked is composed of zero or one interval $[0, v_0]$, zero or several intervals of the kind $[v_1, v_0]$, and zero or one interval of the kind $[v_1, u_0]$, where the multipliers denoted v_1 and v_0 are respectively M1 and 0M multipliers in L_{alg}. Since x_0 is 0-masked, it is in particular masked on one interval $[0, v_0]$. Consequently, x_0 is surrogate feasible and not masked by known solutions on a non-empty union of zero or several (v_0, v_1) open intervals, and zero or one $(v_0, u_0]$ interval. The computation of $CSUB(u)$ for u in the interior of one of these intervals, performed in the context of Algorithm 4, necessarily guarantee to obtain x_0. Otherwise new solutions masking x_0 would be found with their associated critical multipliers, and this contradict the assumption that L_{alg} is the list of known critical multipliers at the end of Algorithm 4.

- If the first kind of interval (v_0, v_1) exists, v_0 and v_1 are respectively a 0M and M1 multiplier in L_{alg}. Hence, there is a 0M-M1 interval (in L_{alg}) in $[v_0, v_1]$ and this 0M-M1 interval is explored during the algorithm.
- If the second kind of interval $(v_0, u_0]$ exists, we prove next that there is a 0M-M1 interval in L_{alg} such that the 0M part is in $[v_0, u_0]$.
  - If there is no 0M multiplier bigger than u_0 in L_{alg}, then since 1 is considered as a M1 multiplier, there is a 0M-M1 interval $u'_0 - 1$ in L_{alg} where u'_0 is the biggest 0M multiplier found.
  - If there is at least one 0M multiplier greater than u_0 in L_{alg}, we denote by $u_{0_{SF}}$ the smallest one. Since $u_0 - u_1$ is a 0M-M1 interval in the list L_{all}, then there exists at least one M1 multiplier in $(v_0, u_{0_{SF}}]$ (at least u_1). We prove next that at least one of these multipliers necessarily belong to L_{alg}.

    Suppose there is no M1 critical multiplier in $(v_0, u_{0_{SF}}] \cap L_{alg}$, we consider u_2 a M1 multiplier in $(v_0, u_{0_{SF}}] \cap L_{all}$ and x_2 one of its associated solution. If there are several possible critical multipliers u_2, we consider one such that x_2 is not masked by any solutions or pair of solutions associated to any other M1 multipliers of the interval $(v_0, u_{0_{SF}}]$. By Proposition 5, the union of intervals on which x_2 is masked is composed of zero or one interval $[u_2, v'_0]$, zero or several intervals of the kind $[v'_1, v'_0]$, and zero or one interval of the kind $[v'_1, 1]$, where the multipliers denoted v'_1 and v'_0 are respectively M1 and 0M multipliers in L_{alg}. Consequently, x_2 is not masked on a non-empty union of zero or several (v'_0, v'_1) intervals, and zero or one $[u_2, v'_1)$ interval. The computation of $CSUB(u)$ for u in the interior of one of these intervals, performed in the context of Algorithm 4, necessarily guarantee to obtain x_2.

    * If the former kind of interval (v'_0, v'_1) exists, v'_0 and v'_1 are respectively a 0M and M1 multiplier in L_{alg}. Hence, there is a 0M-M1 interval in $[v'_0, v'_1] \cap L_{alg}$ and this 0M-M1 interval is explored during the algorithm. Consequently, x_2 and thus u_2 is found.
    * If the later kind of interval $[u_2, v'_1)$ exists, we have $v_0 < u_0 < u_{0_{SF}} < v'_1$. There exists a 0M-M1 interval in $[v_0, v'_1] \cap L_{alg}$. If its 0M part is smaller than u_0 its exploration is the computation of a CSUB in (v_0, u_0), and consequently x_0 (and u_0) is found. Otherwise, the 0M part is greater than u_0 (and thus greater than $u_{0_{SF}}$), consequently u_2 and x_2 are found.

□

Proposition 10. *If $u_0 - u_1$ is a 0M-M1 interval in L_{all}, then there is a M1 multiplier $u_2 \in L_{alg}$ such that no 0M critical multiplier exists in (u_0, u_2).*

Proof. Let $u_0 - u_1$ be a 0M-M1 interval in L_{all}. By Proposition 9, $u_0 \in L_{alg}$.

If u_0 is the greatest 0M multiplier in L_{all}, then there is no 0M multiplier in $(u_0, 1)$ and 1 is (considered as) a M1 multiplier in L_{alg}.

If u_0 is not the greatest 0M multiplier in L_{all}, we denote by u_{0_S} the smallest 0M multiplier in L_{all} such that $u_0 < u_{0_S}$. Since 1 is a M1 critical multiplier, there exists a 0M-M1 interval in L_{all} in $[u_{0_S}, 1]$. According to Proposition 9 the 0M part of this interval is also found by the algorithm. Thus there is at least one 0M multiplier greater than u_0 in L_{alg}, we denote by $u_{0_{SF}}$ the smallest one. Since there is no

0M multiplier in $L_{alg} \cap (u_0, u_{0_{SF}})$, there does not exist any M1 critical multiplier in $L_{all} \cap (u_{0_S}, u_{0_{SF}})$ (otherwise according to Proposition 9 a 0M critical multiplier would be found in $(u_0, u_{0_{SF}})$). Thus the M1 multipliers in $(u_0, u_{0_{SF}}]$ and in $(u_0, u_{0_S}]$ are the same.

We show in the following that there is a M1 multiplier in $L_{alg} \cap (u_0, u_{0_{SF}})$. We consider u_2 a M1 multiplier in L_{all} in $(u_0, u_{0_{SF}}]$ and an associated solution x_2 that is not masked by any other solution (or pair of solutions) associated to a M1 multiplier in $(u_0, u_{0_{SF}}]$. By Proposition 5, the union of intervals on which x_2 is masked is an union of zero or one interval $[u_2, v_0]$, zero or several intervals of the kind $[v_1, v_0]$, and zero or one interval of the kind $[v_1, 1]$, where the multipliers denoted v_1 and v_0 are respectively M1 and 0M multipliers in L_{alg}. Consequently, x_2 is not masked on a non-empty union of zero or several (v_0, v_1) intervals, and zero or one $[u_2, v_1)$ interval.

- If the former kind of interval (v_0, v_1) exists: there exists a 0M-M1 interval in $[v_0, v_1]$ in L_{alg} and it is thus explored during the algorithm.
- If the latter kind of interval $[u_2, v_1)$ exists: since there is no M1 critical multiplier found in $(u_0, u_{0_{SF}}]$ then $u_{0_{SF}} < v_1$. Thus there exists a 0M-M1 interval in $[u_{0_{SF}}, v_1] \subset [u_2, v_1]$ and it is thus explored during the algorithm.

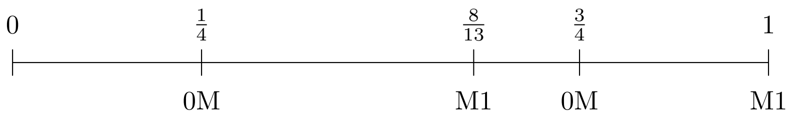
$\square$

Example 1 (continued).

Algorithm 4 is applied on the instance of $(2O2DKP)$ introduced as Example 1. At the beginning *multipliers* contains the M1 multiplier 1. The first bound set to be computed is $CSUB(0) = B_1$. x_1 and x_2 compose this bound set. Then $\frac{1}{4}$ and $\frac{3}{4}$ (of type 0M) are added to *multipliers*.

The only 0M-M1 interval in *multipliers* is $\frac{3}{4} - 1$. Then the next CSUB computed is $CSUB(\frac{3}{4} + \epsilon) = B_2$. This is composed of x_7 and x_8. Thus the multiplier $\frac{8}{13}$ (of type M1) is added to *multipliers* (represented in Figure 8).

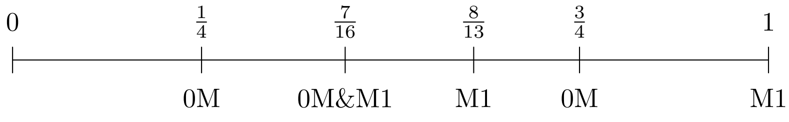
Figure 8: Content of *multipliers* after the computation of B_2



There is two 0M-M1 intervals in *multipliers* now: $\frac{1}{4} - \frac{8}{13}$ and $\frac{3}{4} - 1$. Since $CSUB(\frac{3}{4} + \epsilon)$ has already been computed then we do not compute it again. $CSUB(\frac{1}{4} + \epsilon)$ has not been computed before. So we compute it, it is B_3 composed of x_2 and x_5. The new 0M multiplier $\frac{7}{16}$ is added to *multipliers*.

$\frac{7}{16} - \frac{8}{13}$ is the only new 0M-M1 interval. Since $CSUB(\frac{7}{16} + \epsilon) = B_4$ has not been computed yet, we compute it now. It is composed of x_2 and x_6. The new M1 multiplier $\frac{7}{16}$ is added to *multipliers*. The content of *multipliers* is represented in Figure 9.

Figure 9: Content of *multipliers* after the computation of B_4



The three 0M-M1 intervals found are $\frac{1}{4} - \frac{7}{16}$, $\frac{7}{16} - \frac{8}{13}$ and $\frac{3}{4} - 1$. All of them have already been explored. So the algorithm ends.

In the next section, we present a heuristic derived from the *0M-M1 interval* algorithm, as well as a heuristic from the literature.

6. Methods approximating the OCSUB

6.1. An heuristic based on the 0M-M1 interval *algorithm*

We describe here a method to compute an approximation of the OCSUB, based on the *0M-M1 interval* algorithm and called *0MM1H*. This heuristic works the same way as the *0M-M1 interval* algorithm except that the number of iterations is limited to h. Thus all 0M-M1 interval will generally not be explored. Hence we have to define how we choose the 0M-M1 interval to explore at each iteration. If there are several 0M-M1 intervals $u_0 - u_1$ to explore, then we choose the one for which the u_1 was part of the most 0M-M1 intervals previously explored. In case of equality we choose the lowest u_0. The idea is to explore improving CSUBs immediately.

If there does not exist any 0M-M1 interval to explore, then the method stops even if the maximum number of iterations is not achieved. Indeed the OCSUB is found, any additional computation of CSUB would be useless.

The next section presents a more intuitive heuristic with a lower running time per iteration, since it does not analyze the critical multipliers.

6.2. The SurrogateFamily *heuristic*

The *SurrogateFamily* heuristic (*SF*) comes from the master thesis of Perederieiva (2011). The author has developed a dynamic programming based algorithm to find all efficient solutions of a $2O2DKP$. The upper bound set defined in this algorithm is based on the linear and surrogate relaxation. The resulting upper bound set is obtained by application of Proposition 1, where the upper bound set associated to the linear relaxation and the one based on surrogate relaxations are merged. The aim of *SurrogateFamily* is to find a good trade-off between the computation time and the quality of the bound set. Thus, the output returned by this heuristic is not the OCSUB.

For our numerical experiments, the part related to the linear relaxation is disabled; only the part related to the surrogate relaxation is computed and compared with *0MM1H*. The principle of *SurrogateFamily* is simple. It computes h CSUBs: $\{CSUB(u_1),\ \ldots,\ CSUB(u_h)\}$ where the multipliers $\{u_1, \ldots, u_h\}$ are equidistant in the interval $[0, 1]$. Obviously if we increase the number of multipliers, the upper bound set would be tighter, but the computation cost would be expensive.

In *SurrogateFamily*, the choice of the multipliers is completely static. No information is used to avoid the computation of dominated CSUBs. With a large number of iterations, the OCSUB could be obtained, but *SurrogateFamily* cannot detect this case, and thus it cannot activate a stopping condition in order to save some calculations.

7. Numerical experiments

7.1. Protocol

The numerical experiments aim (1) to show experimentally the performances of proposed algorithms, (2) to observe if an algorithm presents in general better performances than its competitor, (3) to examine the behavior of the algorithm when the number of variables increases, and (4) to note, if it exists, the influence of correlations between the values of the instances on the behavior of the algorithm. To discuss about those factors, five descriptors are elaborated. The three first concern the exact algorithms (*TotalEnumerative* and *0M-M1 interval*),

Descriptor 1. The number of computed CSUBs. A low number shows a fast convergence to the OCSUB.

Descriptor 2. The number of improving CSUBs, *i.e.* CSUBs that improves the current computed bound. A small difference between this number and the total number of computed CSUBs shows the ability of the algorithm to generated relevant CSUBs.

Descriptor 3. CPU time measured to get the OCSUB.

The two last are for the approximation algorithms (*0MM1H* and *SurrogateFamily*).

Descriptor 4. Quality of the approximation. The quality assessment of an approximated bound set will be stated in comparison with the OCSUB. Here, an area measure (called $\mathcal{A}$-metric) is proposed and reported in terms of percentage. More the value is small, more the approximation is close to the OCSUB.

Descriptor 5. CPU time measured to get the approximated OCSUB. A low value is to the advantage of the corresponding algorithm.

h (the number of used directions by the approximation algorithm *i.e.* the maximum number of CSUBs) and ϵ (the smallest difference between two critical multipliers) are the only two parameters to set. The numerical experiments are performed in this paper with $h = \{10, 50, 100\}$ and $\epsilon = 10^{-9}$ (the smaller value manageable by our solver, knowing that the recommended value returned using the computing method described in Appendix A is smaller).

All algorithms are implemented in C++, using the compiler GCC version 4.2.1 on Mac OS X Lion 10.7.5. The tests are performed on Intel Core 2 i7 @2.8 GHz laptop, with 8 Go of RAM memory. We use *Combo* from Martello et al. (1999) to solve single-objective single-dimensional knapsack problems.

7.2. Numerical instances

All algorithms have been evaluated on a dataset composed of 28 instances available (soon) on the MCDMLib[1], and organized in three groups:

Group 1: 6 instances of various sizes, randomly generated with the same generator, without any specific correlation. `ZTL100, ZTL250, ZTL500 and ZTL750` are 4 instances picked from the collection maintained by Zitzler and Laumanns[2]. `ZTL28 and ZTL105` are 2 additional instances derived respectively from ZTL100 and ZTL250, where $\omega_i \approx 0.5 \sum_{j=1}^{n} w_{ij}$, $i = 1, \ldots, m$. The number following ZTL in the name gives the number of variables.

Group 2: 15 correlated instances with 28 variables, introduced by Perederieiva (2011). `A1, A2, A3, A4, D1, D2, D3, D4, kp28W-ZTL, kp28, kp28-2, kp28W, kp28W-Perm, kp28c1W-c2ZTL` and `kp28cW-WZTL` are extention of $2DKP$ instances available on the OR-library[3] where a second objective has been added with respect to several definitions of correlation to obtain a $2O2DKP$ (see Perederieiva (2011) for further details).

Group 3: 7 additional correlated instances with 105 variables, obtained following the rules described for the Group 2. `W7BI-rnd1-1800, W7BI-rnd1-3000, W7BI-tube1-1800, W7BI-tube1-3000, W7BI-tube1-asyn, W7BI-tube2-1800, Wcollage-tube` are summarized in Table 3.

7.3. Comparison between TotalEnumerative *and* 0M-M1 interval

All the results are summarized in Figure 10 (from 10a to 10f). The y-axis is reported on a logarithmic scale for all graphics. The left column reports the measures relating to the CSUBs (descriptor 1 and 2), and the computational times (descriptor 3) on the right column. The line on the top, the middle and the bottom corresponds respectively to instances of Group 1, 2 and 3.

For all three groups of instances, *0M-M1 interval* is always faster than *TotalEnumerative*. The profile of the curves of time are similar between the two algorithms, but with a more expensive running time for *TotalEnumerative*. Also the gap increases with the size of the instance. In considering all the instances, the average ratio "number of improved CSUBs" over "total number of CSUBs calculated" is 25.85% for

[1] http://www.mcdmsociety.org/MCDMlib.html
[2] `http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/`
[3] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html`

Table 3: Information about instances. Column (a) gives the name of the original single-objective $2DKP$. Column (b) indicates the manner for generating the second objective: negatively correlated with the first one or, done according to the method reported in Osorio and Cuaya (2005). Column (c) reports the ratio $\omega_1/\sum_{j=1}^{n} w_{1j}$ for the first dimension (the first constraint). The column (d) is similar to the column (c), for the second dimension.

instance	(a)	(b)	(c)	(d)
W7BI-rnd1-1800	Weing7	Osorio's method	0.48	0.50
W7BI-rnd1-3000	Weing7	Osorio's method	0.80	0.84
W7BI-tube1-1800	Weing7	anti-correlated	0.50	0.50
W7BI-tube1-3000	Weing7	anti-correlated	0.80	0.84
W7BI-tube1-asyn	Weing7	anti-correlated	0.80	0.20
W7BI-tube2-1800	Weing7	anti-correlated	0.48	0.50
Wcollage-tube	Weing1	anti-correlated	0.50	0.50

TotalEnumerative and 53.98% for *0M-M1 interval*. This ratio is always in favor of *0M-M1 interval*. This especially allows *0M-M1 interval* to determine the OCSUB for ZTL500 and ZTL750 instances with less than 700s of CPU time, while *TotalEnumerative* is unable to compute it in less than one hour.

For groups 2 and 3, an obvious correlation appears between the required computation time and the number of CSUBs to compute. On the other hand, there is no particular difficulty for the algorithms to deal with numerical instances presenting various correlations. For some simple instances of Group 2, the difference between *TotalEnumerative* and *0M-M1 interval* on descriptors 1 and 2 is very significant in favor of *0M-M1 interval*. In Group 3, there can be cases where all CSUB are improving. This occurs once for *TotalEnumerative*, while 5 times to *0M-M1 interval*. Also the instance "W7BI-tube1-asyn" requires the computation of a very high number of CSUBs for *TotalEnumerative*, which is observable by an high CPU time.

In conclusion of the analysis, *0M-M1 interval* outranks *TotalEnumerative*, and shows no particular difficulty to deal with instances with correlations. It is the recommended algorithm for computing the OCSUB. However, it remains too expensive in general, to be embedded as a component of an implicit enumeration algorithm aiming to generate a complete set of efficient solutions for a $2O2DKP$. In the state of knowledge, the use of a heuristic version is required.

7.4. $\mathcal{A}$-metric as a quality measure

In order to evaluate the quality of the heuristics, a measure of quality is defined. It aims to compare the search area defined by the OCSUB and the one defined by a heuristic upper bound set. Classically the search area is the zone located on the bottom left of the bound set. If we denote by B the bound set and $\mathcal{A}(V)$ the area of a polyhedra V, a possible measure is: $\mathcal{A}(B - \mathbb{R}^2_{\geqq})$.

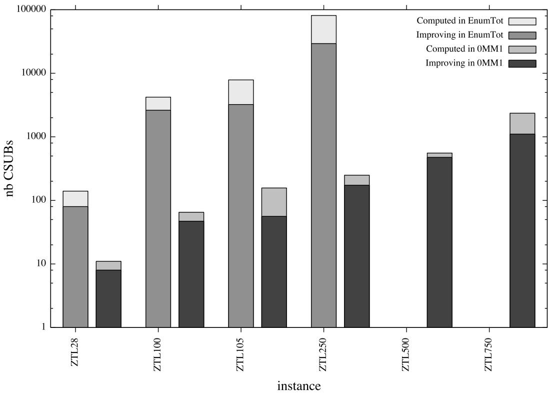
In the experiments, we want to compare the quality of a heuristic bound set H compared to the OCSUB. To do that we compute the difference between $\mathcal{A}(\mathrm{H}{-}\mathbb{R}^2_{\geqq})$ and $\mathcal{A}(OCSUB - \mathbb{R}^2_{\geqq})$.

In order to have the same scale for all instances, we could consider the ratio $\mathcal{A}(\mathrm{H}{-}\mathbb{R}^2_{\geqq})/\mathcal{A}(OCSUB - \mathbb{R}^2_{\geqq})$. However, if the value of the nondominated points can be very large, then a large part of the search area is common to all bound set. Consequently, this indicator will return in most cases values that are near to zero. Such a quality indicator would therefore be useless.

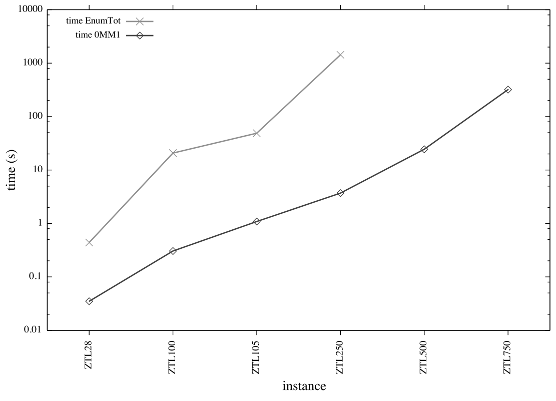
In order to highlight these differences between bound sets, we have to focus on the parts of the search area that are not common and put aside the common part. The OCSUB is composed of b extreme points $\{p_1, \dots, p_b\}$ lexicographically and non increasingly ordered. The search area at the bottom left of p_1 is

Figure 10: Comparison of the *TotalEnumerative* and *0M-M1 interval* algorithms, regarding to the number of CSUBs computed, the number of CSUBs improving and the computational time.

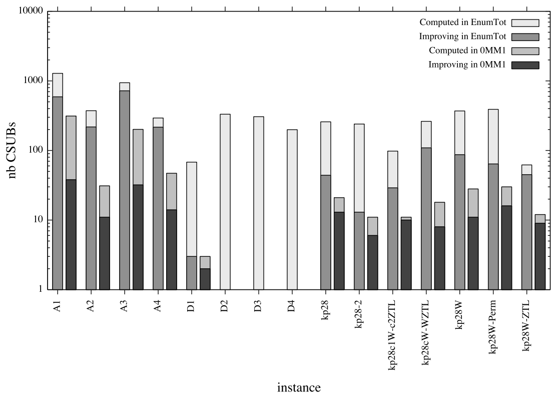
(a) Comparison for the number of CSUBs computed and improving for the Group 1



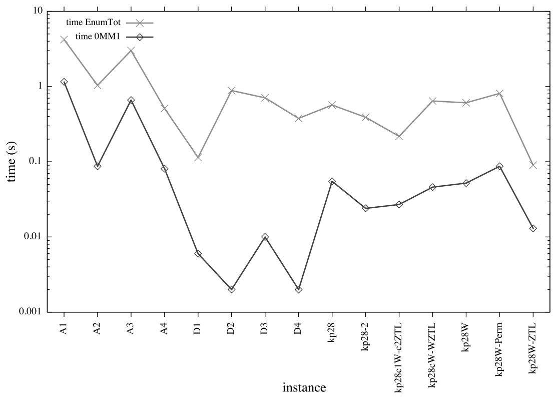
(b) Comparison of running times for the Group 1



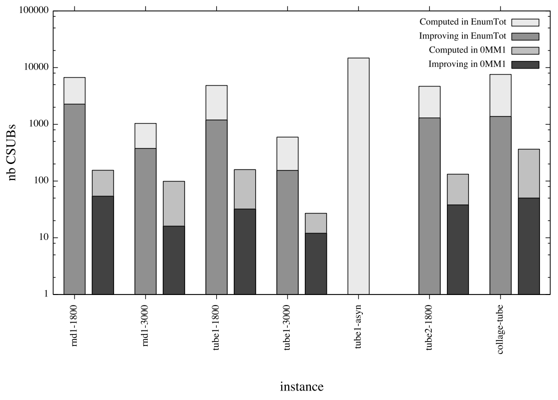
(c) Comparison for the number of CSUBs computed and improving for the Group 2



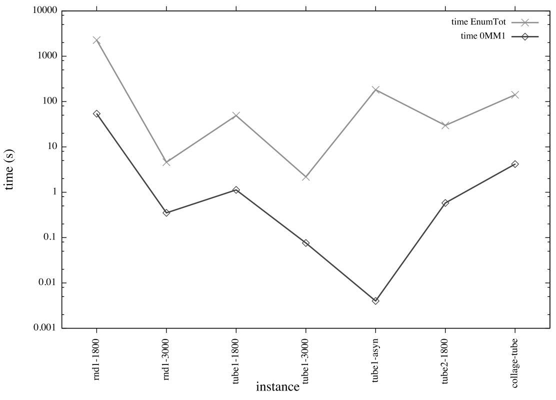
(d) Comparison of running times for the Group 2



(e) Comparison for the number of CSUBs computed and improving for the Group 3

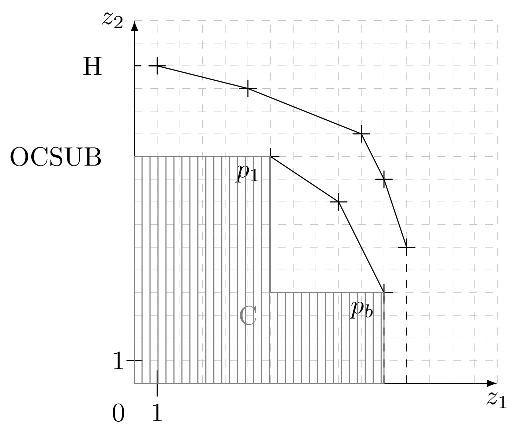


(f) Comparison of running times for the Group 3



common to all the approximations of the OCSUB and to the OCSUB, thus we do not need to consider it in the measure. We same statement holds for p_b. Thus the search area $C := (p_1 - \mathbb{R}^p_{\geqq}) \cup (p_b - \mathbb{R}^p_{\geqq})$ is

Figure 11: Area measure for the comparison



common the search area associated to OCSUB approximations. In the following, we call $\mathcal{A}$*-metric* (Figure 12) the following measure of quality of a heuristic bound set H:

$$\frac{\mathcal{A}(H - \mathbb{R}^p_{\geqq}) - \mathcal{A}(\text{OCSUB} - \mathbb{R}^p_{\geqq})}{\mathcal{A}(\text{OCSUB} - \mathbb{R}^p_{\geqq}) - \mathcal{A}(C)} * 100$$

This quality indicator has the effect to highlight with large values the difference between bound sets.

7.5. Comparison between 0MM1H *and* SurrogateFamily *heuristic*

All the results are summarized in Figure 12 (12a to 12f). The two algorithms are run for $h = \{10, 50, 100\}$. The left column reports the measures relating to the approximated OCSUB (descriptor 4), the y-axis is reported on a logarithmic scale for all the graphics in this column. The right column shows the CPU time (descriptor 5). On the figures in this column, for a same heuristic, the curve at the bottom correspond to $h = 10$ and the one at the top to $h = 100$. The line on the top, the middle and the bottom correspond respectively to instances of Group 1, 2 and 3.

For all three groups of instances, the two algorithms behave the same way in terms of computing time. For a few instances, essentially from Group 2, *0MM1H* appears faster. It is because *SurrogateFamily* has no internal stopping rule to detect that the OCSUB has been found, while *0MM1H* stops when the condition of optimality of the OCSUB is verified.

An approximation of the OCSUB can be obtained with a CPU time smaller than 500 ms for all instances processed. This allows reasonably a use of such heuristics as a component of an implicit enumeration algorithm aiming to compute a complete set of efficient solutions for a $2O2DKP$.

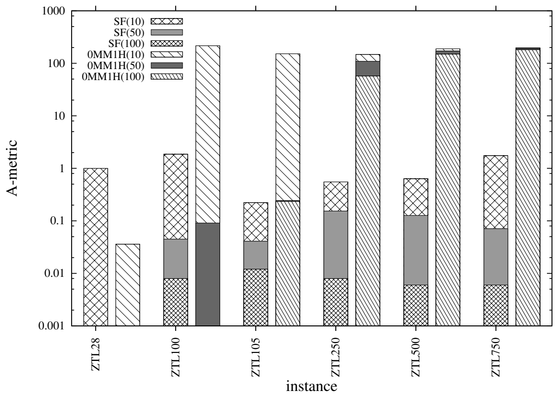
The analysis of the descriptor 4 (figures of the left column) does not point out a winner amongst the algorithms. Indeed, *0MM1H* seems efficient on small instances from of Group 2, while *SurrogateFamily* performs better on the largest instances (example of Group 1, excepted ZTL28). Instances of Group 3 seem to draw the limit of separation of the two algorithms. Depending of the instances, the advantage goes to one or the other method.

The difference of behavior can be explained by the the use of uniformly distributed multipliers in *SurrogateFamily*. Therefore, the obtained bound set is constructed from various CSUBs. In *0MM1H*, the values of the multipliers are determined with the purpose to improve locally the bound set. Thus, if the number of computed CSUBs is low, we obtain a bound set that is very tight locally but not globally. This explanation is confirmed by the results collected for Group 2. *0MM1H* is competitive as soon as the number of computed CSUBs is greater or equal to 50.

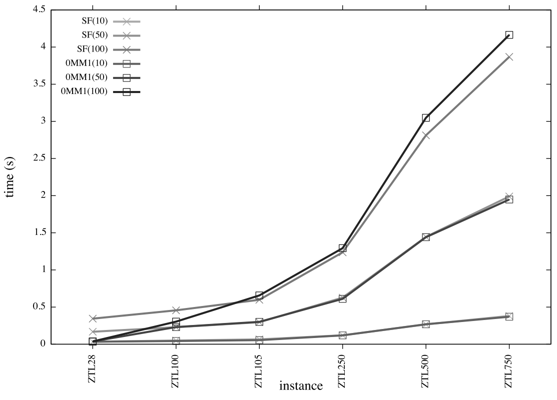
Finally, when *0MM1H* is performed with $h = 100$, the OCSUB are obtained for 16 of the 28 instances, doing fewer iterations than the maximum allowed, and it is verified for all instances of Group 2.

Figure 12: Comparison of the two heuristics *0MM1H* and *SurrogateFamily*, for $h = \{10, 50, 100\}$, regarding to the $\mathcal{A}$-metric and the computational time.

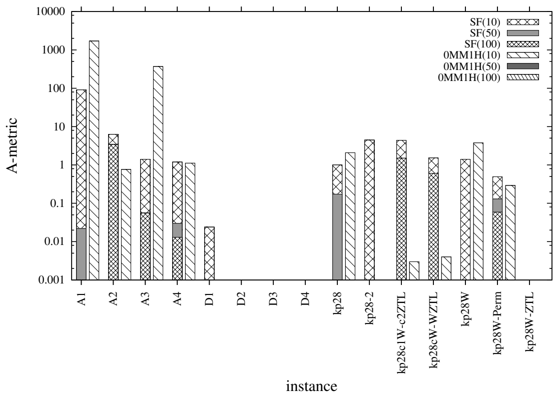
(a) Comparison of $\mathcal{A}$-metric for the Group 1



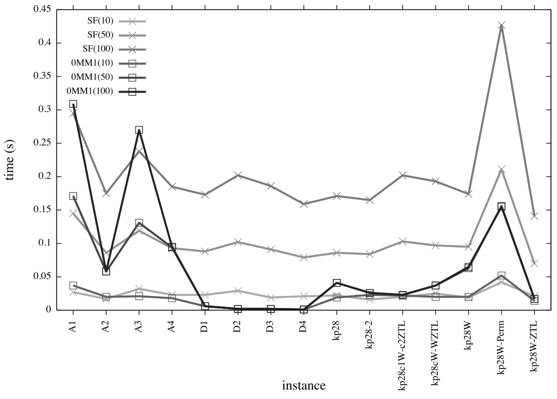
(b) Comparison of running times for the Group 1



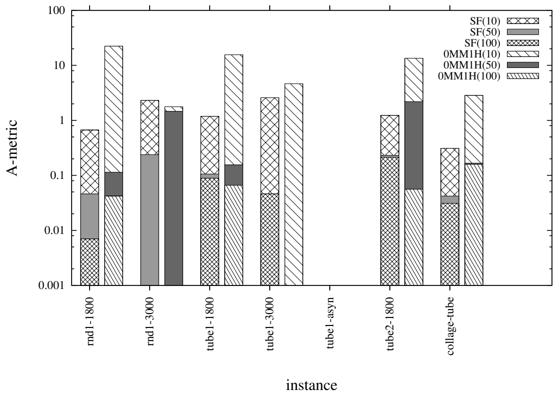
(c) Comparison of $\mathcal{A}$-metric for the Group 2



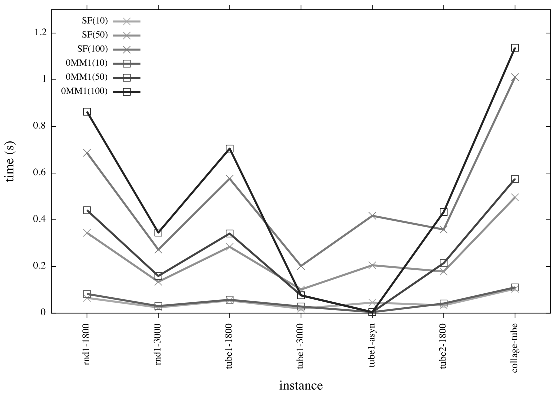
(d) Comparison of running times for the Group 2



(e) Comparison of $\mathcal{A}$-metric for the Group 3



(f) Comparison of running times for the Group 3



In conclusion of the analysis, *SurrogateFamily* takes the advantage on *0MM1H* once the size of instances reaches 100 variables. This shows that the improvement of the computed bound set with *0MM1H* (and also its exact counter-part *0M-M1 interval*) is not as fast as it could be, and could be greatly improved using a suitable set of multipliers as an initialization.

8. Conclusion and on-going work

The optimal convex surrogate upper bound (OCSUB) is the tighter bound set that it is possible to define and to compute with the surrogate relaxation. This paper gives a formal definition of the OCSUB, introduces two exact algorithms for computing the OCSUB, and demonstrates their correctness. A heuristic version is also introduced for a use when the computational time is limited. The numerical experiments clearly demonstrate the effectiveness of the algorithm called *0M-M1 interval* for the computation of the OCSUB, and show that the algorithm is not sensitive to the presence of correlation in the numerical instances.

The OCSUB finds its usage naturally in the context of an implicit enumerative method for solving a bi-objective bi-dimensional binary knapsack problems ($2O2DKP$). For example, OCSUB is ready to be embedded in an evaluation component aiming to prune or not a node of a branch and bound, or a state in dynamic programming. However, it is not surprising to note that the computation of the OCSUB may require a significant CPU time, already for mid-size numerical instances. Due to this fact, and in the state of knowledge, it is currently not reasonable to consider such an usage of the OCSUB. The introduction of a heuristic variant for the computation of the OCSUB is therefore justified. The quality of the resulting approximation measured, even for a short CPU time, shows the practical interest of the heuristic algorithm to compute an approximation of the OCSUB. Thus, it appears as a powerful component to encapsulate in an implicit enumeration method, in particular to solve the 2O2DKP. Currently *0MM1H* does not appear as the best choice for a heuristic computation of OCSUB, as soon as mid-size instances are considered. However there remain several possible improvements that may change favorably this situation.

Two promising prospects allow to expect interesting improvements in terms of the OCSUB computation time, and on the use of the *0MM1H* in an implicit enumeration method. Using an ad-hoc and inexpensive initialization of the surrogate multipliers, *0M-M1 interval* could save a significant amount of time. This first prospect will be considered on base of the paper of Fréville and Plateau (1993) on the (single-objective) dual surrogate problem. This initialization should be profitable to *0MM1H*, (and *0M-M1 interval*) giving it a priori a better distribution of initial multipliers and therefore a better initial coverage of the OCSUB. The second prospect is linked to the use of these algorithms in an implicit enumeration method, following a re-optimization principle, *e.g.* between two levels of the enumeration tree. The idea aims to avoid to repeat computations between two successive OCSUBs (or approximations), while the two nodes in the tree vary only by a small subset of variables changing from a free status to fixed.

Appendix A. Computation of a upper bound on ϵ

In all the algorithms presented here, ϵ is added to critical multipliers u such that $u + \epsilon \neq u$ and there does not exist a critical multiplier u' with $u < u' \leq u + \epsilon$ or $u + \epsilon \leq u' < u$. In those conditions we can find all critical multiplier associated to CSUB.

Our purpose is here to find $\epsilon > 0$ such that it is smaller than the smallest difference between two different critical multipliers.

To find a valid value for ϵ, we consider u and u' two different critical multipliers, such that the difference between u and u' is the smallest difference between two different critical multipliers. x is a solution associated to u and x' is a solution associated to u'. We have shown in Section 3.1 that $u = \frac{\omega_2 - w_2(x)}{w_1(x) - w_2(x) - \omega_1 + \omega_2}$. To simplify the notation we denote $p_i = w_i(x), q_i = w_i(x')$ for $i = 1, 2$. We have therefore

$$
\begin{aligned}
|u-u'| &= \left| \frac{\omega_2 - p_2}{p_1 - p_2 - \omega_1 + \omega_2} - \frac{\omega_2 - q_2}{q_1 - q_2 - \omega_1 + \omega_2} \right| \\
&= \frac{|(\omega_2 - p_2)(q_1 - q_2 - \omega_1 + \omega_2) - (\omega_2 - q_2)(p_1 - p_2 - \omega_1 + \omega_2)|}{|(p_1 - p_2 - \omega_1 + \omega_2)(q_1 - q_2 - \omega_1 + \omega_2)|}.
\end{aligned}
$$

If the numerator equals 0 then $u = u'$, and we make the assumption that both multipliers are different. Its value is then greater or equal to 1. Now we search the maximum value of the denominator, denoted $|D|$.

$$
\begin{aligned}
D =& (p_1 - p_2 - \omega_1 + \omega_2)(q_1 - q_2 - \omega_1 + \omega_2) \\
D =& p_1q_1 + p_1\omega_2 + p_2q_2 + p_2\omega_1 + \omega_1q_2 + \omega_1^2 + \omega_2q_1 + \omega_2^2 \\
&- (p_1q_2 + p_1\omega_1 + p_2q_1 + p_2\omega_2 + \omega_1q_1 + 2\omega_1\omega_2 + \omega_2q_2) \\
D \leq& \max(p_1q_1 + p_1\omega_2 + p_2q_2 + p_2\omega_1 + \omega_1q_2 + \omega_1^2 + \omega_2q_1 + \omega_2^2) \\
&- \min(p_1q_2 + p_1\omega_1 + p_2q_1 + p_2\omega_2 + \omega_1q_1 + 2\omega_1\omega_2 + \omega_2q_2)
\end{aligned}
$$

Since p_1 is the weight of the solution on the first dimension, we have $0 \leq p_1 \leq M_1$ with $M_1 = \sum_{j=0}^{n} w_{1j}$. Similarly $0 \leq q_1 \leq M_1$. Symmetrically $M_2 = \sum_{j=0}^{n} w_{2j}$, $0 \leq p_2 \leq M_2$ and $0 \leq q_2 \leq M_2$.

We can thus deduce that:

$$D \leq M_1^2 + 2M_1\omega_2 + M_2^2 + 2M_2\omega_1 + \omega_1^2 + \omega_2^2 - 2\omega_1\omega_2 = D_{max}$$

and similarly

$$D \geq \omega_1^2 + \omega_2^2 - 2\omega_1\omega_2 - 2M_1M_2 - 2M_1\omega_1 - 2M_2\omega_2 = D_{min}$$

Thus $|D| \leq \max(|D_{min}|, |D_{max}|)$ and $\epsilon < \frac{1}{\max(|D_{min}|,|D_{max}|)}$.
$\frac{1}{\max(|D_{min}|,|D_{max}|)}$ is an upper bound on ϵ.

References

Aneja, Y. P., Nair, K. P. K., 1979. Bicriteria Transportation Problem. Management Science 25 (1), 73–78.

Bazgan, C., Hugot, H., Vanderpooten, D., 2009. Solving efficiently the 0-1 multi-objective knapsack problem. Computers and Operations Research 36, 260–279.

Boyer, V., 2007. Contribution à la programmation en nombres entiers. Ph.D. thesis, Université de Toulouse.

Boyer, V., Elkihel, M., Baz, D. E., 2009. Heuristics for the 01 multidimensional knapsack problem. European Journal of Operational Research 199 (3), 658 – 664.

Clausen, T., Hjorth, A. N., Nielsen, M., Pisinger, D., 2010. The off-line group seat reservation problem. European Journal of Operational Research 207 (3), 1244 – 1253.

da Silva, C. G., Clímaco, J. C. N., Figueira, J. R., 2004. A scatter search method for the bi-criteria multi-dimensional {0, 1}-knapsack problem using surrogate relaxation. Journal of Mathematical Modelling Algorithms 3 (3), 183–208.

Delort, C., Spanjaard, O., 2010. Using bound sets in multiobjective optimization: application to the biobjective binary knapsack problem. In: Proceedings of the 9th international conference on Experimental Algorithms. SEA'10. Springer-Verlag, Berlin, Heidelberg, pp. 253–265.

Ehrgott, M., 2005. Multicriteria Optimization, 2nd Edition. Springer, Berlin - Heidelberg.

Ehrgott, M., Gandibleux, X., 2001. Bounds and bound sets for biobjective combinatorial optimization problems. In: Köksalan, M., Zionts, S. (Eds.), Multiple Criteria Decision Making in the New Millenium. Vol. 507 of Lecture Notes in Economics and Mathematical Systems. Springer Verlag, Berlin, pp. 242–253.

Ehrgott, M., Gandibleux, X., 2007. Bound sets for biobjective combinatorial optimization problems. Computers and Operations Research 34 (9), 2674 – 2694.

Fleszar, K., Hindi, K. S., 2009. Fast, effective heuristics for the 01 multi-dimensional knapsack problem. Computers and Operations Research 36 (5), 1602 – 1607.

Florios, K., Mavrotas, G., Diakoulaki, D., 2010. Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. European Journal of Operational Research 203 (1), 14–21.

Fréville, A., 2004. The multidimensional 0–1 knapsack problem: An overview. European Journal of Operational Research 155 (1), 1 – 21.

Fréville, A., Plateau, G., 1993. An exact search for the solution of the surrogate dual of the 0-1 bidimensional knapsack problem. European Journal of Operational Research 68 (3), 413–421.

Gandibleux, X., Perederieieva, O., 2011. Some observations on the bi-objective 01 bi-dimensional knapsack problem. IFORS 2011 (19th Triennial Conference of the International Federation of Operational Research Societies). 10–15 July 2011, Melbourne, Australia.

Geoffrion, A. M., 1968. Proper Efficiency and the Theory of Vector Maximization. Journal of Mathematics, Analysis and Applications 22 (3), 618–630.

Glover, F., 1975. Surrogate Constraint Duality in Mathematical Programming. Operations Research 23, 434–451.

Ishibuchi, H., Hitotsuyanagi, Y., Tsukamoto, N., Nojima, Y., 2009. Implementation of multiobjective memetic algorithms for combinatorial optimization problems: A knapsack problem case study. In: Goh, C.-K., Ong, Y.-S., Tan, K. (Eds.), Multi-Objective Memetic Algorithms. Vol. 171 of Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 27–49.

Jaszkiewicz, A., 2004. On the computational efficiency of multiple objective metaheuristics. the knapsack problem case study. European Journal of Operational Research 158 (2), 418 – 433.

Jorge, J., 2010. Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires. Ph.D. thesis, Université de Nantes.
URL `http://tel.archives-ouvertes.fr/tel-00488215`

Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack problems. Springer.

Lust, T., Teghem, J., 2012. The multiobjective multidimensional knapsack problem: a survey and a new approach. International Transactions in Operational Research 19 (4), 495–520.

Martello, S., Pisinger, D., Toth, P., 1999. Dynamic programming and strong bounds for the 0-1 knapsack problem. Management Science 45 (3), 414–424.

Martello, S., Toth, P., 1990. Knapsack Problems : Algorithms and Computer Implementations. John Wiley & sons.

Mavrotas, G., Figueira, J. R., Antoniadis, A., 2011. Using the idea of expanded core for the exact solution of bi-objective multi-dimensional knapsack problems. Journal of Global Optimization 49 (4), 589–606.

Osorio, M. A., Cuaya, G., 2005. Hard problem generation for MKP. In: Proceedings of the Sixth Mexican International Conference on Computer Science. ENC '05. IEEE Computer Society, Washington, DC, USA, pp. 290–297.

Perederieiva, O., 2011. Bound sets for bi-objective bi-dimensional knapsack problem. Master thesis, Université de Nantes.

Pisinger, D., 1994. A minimal algorithm for the 0-1 knapsack problem. Operations Research 45, 758–767.

Puchinger, J., Raidl, G. R., Pferschy, U., 2010. The multidimensional knapsack problem: Structure and algorithms. INFORMS Journal on Computing 22 (2), 250–265.

Tricoire, F., 2012. Multi-directional local search. Computers and Operations Research 39 (12), 3089 – 3101.

Ulungu, E. L., Teghem, J., 1997. Solving multi-objective knapsack problem by a branch-and-bound procedure. In: Clímaco, J. (Ed.), Multicriteria Analysis. Springer Verlag, Berlin, pp. 269–278.

Varnamkhasti, M. J., 2012. Overview of the algorithms for solving the multidimensional knapsack problems. Advanced Studies in Biology 4 (1-4), 37 – 47.

Visée, M., Teghem, J., Pirlot, M., Ulungu, E. L., 1998. Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem. Journal of Global Optimization 12, 139–155.

Wilbaut, C., Hanafi, S., Salhi, S., 2008. A survey of effective heuristics and their application to a variety of knapsack problems. IMA Journal of Management Mathematics 19 (3), 227–244.

Zitzler, E., Thiele, L., 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3 (4), 257–271.